7-2021

Privacy-Preserving Cloud-Assisted Data Analytics

Wei Bao
University of Arkansas, Fayetteville

Privacy-Preserving Cloud-Assisted Data Analytics

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy in Engineering with a concentration in Computer Science

by

Wei Bao
Huazhong University of Science & Technology
Bachelor of Science in Industrial Engineering, 2012
University of Arkansas
Master of Science in Industrial Engineering, 2014

July 2021
University of Arkansas

This dissertation is approved for recommendation to the Graduate Council

---

Qinghua Li, Ph.D.
Dissertation Chair

Xintao Wu, Ph.D.
Committee Member

Brajendra Nath Panda, Ph.D.
Committee Member

Jingxian Wu, Ph.D.
Committee Member

ABSTRACT

Nowadays industries are collecting a massive and exponentially growing amount of data that can be utilized to extract useful insights for improving various aspects of our life. Data analytics (e.g., via the use of machine learning) has been extensively applied to make important decisions in various real world applications. However, it is challenging for resource-limited clients to analyze their data in an efficient way when its scale is large. Additionally, the data resources are increasingly distributed among different owners. Nonetheless, users' data may contain private information that needs to be protected.

Cloud computing has become more and more popular in both academia and industry communities. By pooling infrastructure and servers together, it can offer virtually unlimited resources easily accessible via the Internet. Various services could be provided by cloud platforms including machine learning and data analytics.

The goal of this dissertation is to develop privacy-preserving cloud-assisted data analytics solutions to address the aforementioned challenges, leveraging the powerful and easy-to-access cloud. In particular, we propose the following systems.

To address the problem of limited computation power at user and the need of privacy protection in data analytics, we consider geometric programming (GP) in data analytics, and design a secure, efficient, and verifiable outsourcing protocol for GP. Our protocol consists of a transform scheme that converts GP to DGP, a transform scheme with computationally indistinguishability, and an efficient scheme to solve the transformed DGP at the cloud side with result verification. Evaluation results show that the proposed secure outsourcing protocol can achieve significant time savings for users.

To address the problem of limited data at individual users, we propose two distributed learning systems such that users can collaboratively train machine learning models without losing privacy. The first one is a differentially private framework to train logistic regression

models with distributed data sources. We employ the relevance between input data features and the model output to significantly improve the learning accuracy. Moreover, we adopt an evaluation data set at the cloud side to suppress low-quality data sources and propose a differentially private mechanism to protect user's data quality privacy. Experimental results show that the proposed framework can achieve high utility with low quality data, and strong privacy guarantee.

The second one is an efficient privacy-preserving federated learning system that enables multiple edge users to collaboratively train their models without revealing dataset. To reduce the communication overhead, we select well-aligned and large-enough magnitude gradients for uploading which leads to quick convergence. To minimize the noise added and improve model utility, each user only adds a small amount of noise to his selected gradients, encrypts the noise gradients before uploading, and the cloud server will only get the aggregate gradients that contain enough noise to achieve differential privacy. Evaluation results show that the proposed system can achieve high accuracy, low communication overhead, and strong privacy guarantee.

In future work, we plan to design a privacy-preserving data analytics with fair exchange, which ensures the payment fairness. We will also consider designing distributed learning systems with heterogeneous architectures.

ACKNOWLEDGMENTS

First and foremost, I would like to thank my advisor Prof. Qinghua Li. I greatly appreciate all his contributions of time, idea, and support to make my Ph.D. study productive. He is always available and supportive with amazing insights and advice. I am so fortunate to have had the opportunity to work with him in the past few years.

My gratitude also goes to other members of my dissertation and advisory committee, including Prof. Xintao Wu, Prof. Brajendra Nath Panda, and Prof. Jingxian Wu. I would like to thank them for their constant support, feedback, and encouragement during my PhD study.

Finally, on a personal note, I express my deepest gratitude to my beloved family for their unconditional love, encouragement, and support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF PUBLISHED PAPERS

1. **Chapter 2: Wei Bao**, and Qinghua Li, "Efficient Privacy-Preserving Outsourcing of Large-Scale Geometric Programming," *in the IEEE Symposium on Privacy-Aware Computing (PAC), 2018.*

1 Introduction

In recent years, the data collected every day via social media and networks, mobile phones, and all other channels has been growing exponentially. These data would provide huge amount of potential useful information and insights. Data analytics has been widely used to extract insights from massive amount of data via machine learning techniques, such as deep neutral networks, decision tree, and support vector machine. For example, hospitals have been analyzing patients' medical records to provide most cost-effective treatment plan [1]; financial institutes also perform frequent data analytics process to detect fraudulent transactions and illegal behaviors [2]; smart grid systems employ various data anayltics tools to monitor the power transmission process [3]. Mathematical optimization techniques have been employed frequently to improve performance of fundamental machine learning algorithms. For example, authors in [4] use convex optimization to improve neural network model accuracy. However, it is very challenging for individuals and small corporations to process large-scale data due to limited resources. Moreover, data resources are increasingly distributed and stored by different owners. How to efficiently analyze distributed data poses challenges.

Cloud computing has been widely adopted in both academia and industry communities. With Internet access, it provides unlimited resources through the on-demand business model, which helps solve computation-extensive tasks for resource-limited users. Also, the cloud service can be easily reached through computers and mobile phones. That enables a variety of services to be provided by the cloud, including cloud-facilitated data analytics.

Although cloud computing is promising to help solve large-scale mathematical optimization problems in data analytics, outsourcing mathematical optimization problems to

the cloud may pose security and privacy concerns. The first concern is data privacy for both input data and model output. There may exist sensitive information in the problem formulation and model results, thus we need to protect their privacy during outsourcing process. To protect data against potential leakage, one can encrypt problem formulation before outsourcing to the cloud. Another concern is the verifiability of results returned from the cloud server. Since data operations on the cloud side remains unclear to the users, it is necessary to verify the returned results. For example, the cloud server might not follow our proposed algorithm to save computing resources. If the cloud sever is under attack during the computation period or suffers from system failures, it might also return incorrect results. In other words, the cloud server is considered malicious and the outsourcing algorithm should be able to check the correctness of the returned results.

In addition, machine learning models for data analytics usually require a vast amount of data, which may prevent individual users from obtaining high-accuracy learning models based on their own data. One feasible solution would be that users collaboratively train the machine learning models under the coordination of a cloud server. However, the users may own private data and cannot disclose the data to the cloud server or other users. For example, hospitals and healthcare institutes may be incentivized to collaboratively train machine learning models for better diagnosis results, but health data is sensitive and should not be shared with any third party. Moreover, communication overhead in the distributed learning setting could also be a potential issue for edge/mobile devices that have limited bandwidth.

1.1 Overview of This Dissertation

The goal of this dissertation is to design privacy-preserving cloud-assisted mathematical optimization and machine learning systems for data analytics. For this purpose, we

propose several cloud-assisted systems.

1.1.1 Privacy-Preserving Cloud-Assisted Mathematical Optimization for Data Analytics

Mathematical optimization has been widely adopted in data analytics. For example, Khosravi et al. [4] proposed to use Geometric Programming (GP) to learn a naive Bayes distribution for improving the performance of logistic regression classifier with missing features. Solving mathematical optimization for data analytics is requiring extensive computation power when the data scale is large. Some recent works [5, 6, 7] have been proposed to solve large scale mathematical optimization by outsourcing the problem to the cloud server, and homomorphic encryption techniques were adopted to protect their data privacy. However, those techniques would usually introduce high computation cost and complex encryption/decryption operations. Moreover, the users cannot verify the correctness of returned results from the cloud server.

To address these issues, we design an efficient and privacy-preserving system for outsourcing GP problems to the cloud server [8], which has not been studied before. In particular, we consider a general GP problem. The GP is first converted to a convex dual geometric problem (DGP) by variable substitutions and the Lagrange dual method. Next, the user transforms (i.e., encrypts) the DGP through multiplying the decision variable and constraints by random sparse matrices. We prove that the transformed DGP is computationally indistinguishable from the DGP both in value and in structure. Then based on the dual problem theory and the gradient projection method, the cloud solves the transformed DGP, and sends the result to the user, who can then efficiently derive the solution to its original GP and verify the solution. The scheme protects the user's privacy by letting the cloud operate on the transformed DGP, rather than any original problem formulations. We

implement the system with both Amazon Elastic Compute Cloud (EC2) and a laptop to evaluate performance of the designed outsourcing protocol. Experimental results show that the proposed secure outsourcing system can achieve significant time savings for users.

1.1.2 Privacy-Preserving Cloud-Assisted Machine Learning for Data Analytics

To benefit from other users' data in machine learning while protecting data privacy, we propose two privacy-preserving cloud-assisted systems that each achieves differential privacy. The first system is *Privacy-Preserving Cloud-Assisted Distributed Logistic Regression* that has high learning accuracy and is robust against low quality data of participating users. To achieve differential privacy during training, noise is usually added to the gradients before uploading to cloud server. However, model accuracy would be affected by the noisy gradients at each iteration. Our basic idea is to more wisely add noise by exploring the relevance connection between model output and input features. That allows us to add less noise to objective coefficients with high relevance, and vice-versa. This way, the model accuracy could be improved significantly. Specifically, first each user computes the magnitude of relevance between the learning output and the input data features based on layer-wise relevance propagation [9]. Then, the local logistic regression learning objective is approximated with function of polynomial terms. Based on magnitudes of relevance for each data feature, different carefully-crafted noises are injected to the different coefficients of the polynomial objective function for achieving differential privacy. In addition, some users may hold low-quality data possibly due to inaccurate data collection processes, which may impact the effectiveness of the trained model in distributed learning. Thus, we utilize an evaluation dataset on the cloud side to measure how good each users data quality is when the local parameters are uploaded to the cloud server and filter out low-quality data. Since low-quality data are not included in the aggregation process, the learned model performance would not

be significantly impacted. Moreover, the global parameters are updated in a way that preserves differential privacy, and user's data quality privacy is also protected. This process iterates until the training converges.

The second system is *Privacy-Preserving Cloud-Assisted Efficient Federated Learning*. Federated Learning uses some form of distributed stochastic gradient descent (SGD) and requires a cloud server to coordinate the training process. It usually involves many iterations and incurs high communication cost. To reduce the communication cost, our basic idea is to select gradients more effectively to achieve faster convergence of training. In particular, each user trains on his local dataset and select the gradients that are *aligned* with global model gradient tendency, as some local updates may not contribute to the model convergence due to non-IID (independent and identically distributed) data among users. Gradient magnitude is also considered as another selection criteria as large magnitude usually means more impact to model training. By excluding those less consistent gradient uploads, the global model updates would converge much faster as those uploads may contribute nothing to or diverge the global updates. For privacy protection, we adopt the basic approach of adding noise to gradients to achieve differential privacy, and try to minimize the noise and improve learning accuracy by leveraging cryptography techniques. Existing work on this topic either adds too much noise at each user [10] or relies on the untrusted server to add noise. Different from them, in our solution, each client will first add a small amount of noise to perturb the gradients uploading to cloud server. Then, homomorphic encryption is adopted to encrypt the noisy gradients before uploading to the cloud server. Finally, the cloud server decrypts the sum of the noisy gradients and updates the global model parameters without learning anything else. The aggregate amount of noise in the sum is minimal but sufficient for differential privacy.

1.2 Organization

The remainder of the dissertation is organized as follows. Chapter 2 presents an efficient privacy-preserving outsourcing protocol to solve large-scale Geometric Programming. Chapter 3 introduces a collaborative privacy-preserving logistic regression learning system. Chapter 4 presents a privacy-preserving efficient federated learning system. Chapter 5 concludes the dissertation and discusses future work.

Bibliography

[1] K. Y. Ngiam and W. Khor, "Big data and machine learning algorithms for health-care delivery," *The Lancet Oncology*, vol. 20, no. 5, pp. e262–e273, 2019.

[2] T. U. Rehman, M. S. Mahmud, Y. K. Chang, J. Jin, and J. Shin, "Current and future applications of statistical machine learning algorithms for agricultural machine vision systems," *Computers and electronics in agriculture*, vol. 156, pp. 585–605, 2019.

[3] E. Liberty, Z. Karnin, B. Xiang, L. Rouesnel, B. Coskun, R. Nallapati, J. Delgado, A. Sadoughi, Y. Astashonok, P. Das, *et al.*, "Elastic machine learning algorithms in amazon sagemaker," in *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pp. 731–737, 2020.

[4] P. Khosravi, Y. Liang, Y. Choi, and G. V. d. Broeck, "What to expect of classifiers? reasoning about logistic regression with missing features," *arXiv preprint arXiv:1903.01620*, 2019.

[5] Q. Wang, J. Wang, S. Hu, Q. Zou, and K. Ren, "Sechog: Privacy-preserving outsourcing computation of histogram of oriented gradients in the cloud," in *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*, pp. 257–268, ACM, 2016.

[6] C. Hu, A. Alhothaily, A. Alrawais, X. Cheng, C. Sturtivant, and H. Liu, "A secure and verifiable outsourcing scheme for matrix inverse computation," in *INFOCOM 2017-IEEE Conference on Computer Communications, IEEE*, pp. 1–9, IEEE, 2017.

[7] X. Lei, X. Liao, T. Huang, and F. Heriniaina, "Achieving security, robust cheating resistance, and high-efficiency for outsourcing large matrix multiplication computation to a malicious cloud," *Information sciences*, vol. 280, pp. 205–217, 2014.

[8] W. Bao and Q. Li, "Efficient privacy-preserving outsourcing of large-scale geometric programming," in *2018 IEEE Symposium on Privacy-Aware Computing (PAC)*, pp. 55–63, IEEE, 2018.

[9] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek, "On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation," *PloS one*, vol. 10, no. 7, p. e0130140, 2015.

[10] M. Hao, H. Li, G. Xu, S. Liu, and H. Yang, "Towards efficient and privacy-preserving federated deep learning," in *ICC 2019-2019 IEEE International Conference on Communications (ICC)*, pp. 1–6, IEEE, 2019.

2 Privacy-Preserving Outsourcing of Large-Scale Geometric Programming to the Cloud

2.1 Introduction

Mathematical optimization has found applications in various data analytics areas, such as computer science [1], signal processing [2], and economics [3]. If the objective function is geometric and the feasible region is constrained to a system of linear equalities and inequalities, it is called Geometric Programming (GP), one of the most widely used mathematical optimization. A number of works have been proposed to adopt GP in data mining tasks. For instance, GP is utilized to learn a naive Bayes distribution that improves the performance of a given logistic regression classifier and can efficiently output expected predictions with missing features [4]. The work in [5] employed GP to obtain the optimal power and resource allocation in blockchain computation tasks. However, solving GP requires many computing resources when its scale is large. Thus, it is attractive for a client with low computing capability to outsource large-scale GP problems to the cloud.

Although outsourcing to the cloud allows a client to solve large-scale GP problems, it also brings some new issues [6, 7]. Privacy is the first issue to be handled. Since both the outsourced tasks and the results to these tasks may contain sensitive information that the client does not expect to be exposed to the cloud. To ensure the input privacy (i.e., secrecy of the original GP problem) is not breached, the client has to encrypt the original problem before uploading it to the cloud. The output privacy should also be protected, which means the cloud should not be able to infer the solution to the original GP problem.

Verifiability is the next issue to be considered. The client needs to verify whether the result returned is correct or not. The cloud may return a random result to save computing

resources when the outsourced task is highly resource-consuming. Even though the cloud performs faithfully, some inevitable hardware and software bugs in cloud may also lead to an incorrect result. Thus without verification, the correctness of the result returned by the cloud cannot be guaranteed.

Lastly, efficiency is also an issue that needs to be addressed. It requires that the overhead of the client should be substantially reduced when outsourcing is chosen. Furthermore, the amount of computation performed by the cloud is comparable to the overload of solving the original problem.

In this chapter, we develop an efficient and privacy-preserving algorithm for outsourcing GP problems [8]. Specifically, we consider a general GP problem. Due to the characteristics of GP, first the GP is converted to a convex dual geometric problem (DGP) by variable substitutions and the Lagrange dual method. Next, the client transforms (i.e., encrypts) the DGP through multiplying the decision variable and constraints by random sparse matrices. We show that the transformed DGP is computationally indistinguishable from the DGP both in value and in structure. Then based on the dual problem theory and the gradient projection method, the cloud solves the transformed DGP, and sends the result to the client, who can then efficiently derive the solution to its original GP and verify the solution. The algorithm protects the client's privacy by letting the cloud operate on the transformed DGP, rather than any original problem matrices.

The main contributions of this chapter are summarized as follows:

- To the best of our knowledge, it is the first privacy-preserving solution for outsourcing GP problems to the cloud. It consists of a transform scheme that conveys GP to DGP, a transform scheme that protects the DGP, and a scheme to solve the transformed DGP at the cloud side.

- We formally prove that the transformed DGP problem can protect the client's data privacy. In particular, the transformed DGP has the property of computationally indistinguishability.

- We implement the proposed solution on the Amazon EC2 platform and a laptop. Experimental results show that the proposed secure outsourcing mechanism can achieve significant time savings for the client.

The rest of the chapter is organized as follows. Section 2.2 presents problem formulation. Section 2.3 introduces the secure transformation scheme. Section 2.4 describes the DGP problem. Section 2.5 presents in detail the algorithm for solving the outsourced DGP. Performance evaluation is presented in Section 2.6. Section 2.7 reviews related work. Section 2.8 concludes this chapter.

2.2 Problem Formulation

2.2.1 Geometric Problem Formulation

GP is a class of mathematical optimization and has the following form:

$$\begin{aligned} \text{Minimize} \quad & f_0(\mathbf{x}) = \sum_{j=1}^{N_0} c_j \prod_{i=1}^{n} x_i^{a_{ij}} \\ \text{subject to} \quad & g_k(\mathbf{x}) = \sum_{j=N_{k-1}+1}^{N_k} c_j \prod_{i=1}^{n} x_i^{a_{ij}} \leq 1, \ k = 1, \ldots, m \\ & \mathbf{x} > \mathbf{0} \end{aligned} \tag{2.1}$$

where $\mathbf{x} = (x_1, x_2, \ldots, x_n)^t$ is the optimization variable, and N_k, for $k = 0, 1, \ldots, m$, and c_j, for $j = 1, 2, \ldots, N_m$ represent the number of terms in each function and term coefficients, respectively. Problem (2.1) is said to be a geometric programming problem because both

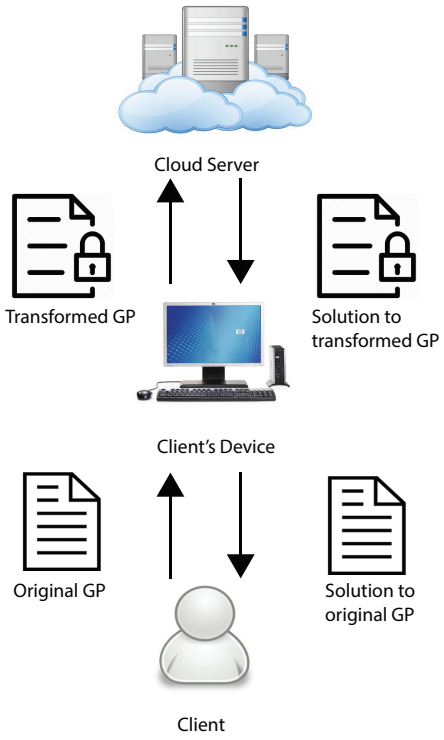


Figure 2.1: System architecture for outsourcing GP

the objective function and constraint functions can be expressed as the sum of posynomial terms [9].

GP problems arise frequently in engineering applications. For example, in a power control problem, each decision variable x_i represents the positive transmitting power level, and the interface power of each transmitter/receiver pair can be expressed as posynomial terms. This problem can be formulated as a GP problem where decision variables are subject to practical constraint functions. Another example is semiconductor device operations. In particular, the objective function is to choose the doping profile to minimize the base transit time, while the doping profile value is bounded and consists of posynomial terms. Obviously, this problem can also be formulated as a GP problem.

2.2.2 System Architecture

As shown in Fig. 2.1, we consider an asymmetric two-party computing architecture, where a local client is resource limited while a remote cloud server has abundant computing resources. The client is unable to solve the original GP problem with local computational resources in an acceptable amount of time. Thus, the client outsources the GP problem to the cloud after making certain transformations to it (it is called transformed GP after transformations). Then, the cloud server solves the transformed GP and sends the solution of the transformed GP back to the client, who will verify and decrypt the solution for the original GP problem.

2.2.3 Threat Model

We assume a malicious cloud server. In particular, the cloud attempts to learn the client's original GP problem from the outsourced problem and the returned results of its own computations. Additionally, the cloud may not follow the proposed protocol and return incorrect results.

To securely outsource the computation of the GP problem, we adopt the concept of computational indistinguishability under a chosen plaintext attack (CPA) [10]. In a matrix, we notice that the elements' values and positions both carry private information. In the following, we formally define computational indistinguishability under a CPA for these two types of private information, respectively.

We first present the definition of a pseudorandom function as follows, which will be employed to perform matrix transformations with CPA security.

Definition 1. Let $X = \{X_n\}_{n\in N}$ be a probability ensemble and $Y = \{Y_n\}_{n\in N}$ be a truly random function. We say X is a pseudorandom function if for all probabilistic

polynomial-time distinguishers D, there exists a negligible function μ such that

$$|Pr[D(X_n) = 1] - Pr[D(Y_n) = 1]| \leq \mu \tag{2.2}$$

This definition can be extended to the case where a distinguisher D has access to multiple elements of the vectors X and Y, i.e., when comparing two matrices.

Definition 2. Let $\mathbf{R} \in R^{m \times n}$ be a random matrix with elements in its jth column sampled from a uniform distribution with interval $[-R_j, R_j]$ $\forall j \in [1, n]$. Matrices $\mathbf{R}$ and $\mathbf{G} \in R^{m \times n}$ are computationally indistinguishable if for any probabilistic polynomial time distinguisher D there exists a negligible function β such that

$$|Pr[D(g_{ij}) = 1] - Pr[D(r_{ij}) = 1]| \leq \beta \tag{2.3}$$

where $i \in [1, m], j \in [1, n], g_{ij}$ is the element in the ith row and jth column of $\mathbf{G}$, and r_{ij} is the element in the ith row and jth column of $\mathbf{R}$. Distinguisher D outputs 1 when it finds out g_{ij} is not chosen from matrix $\mathbf{G}$ and 0 otherwise.

2.3 A Privacy-Preserving Transformation Scheme

To securely outsource a GP problem to the cloud, the client must first encrypt the problem by performing certain computations. In this section, we describe a privacy-preserving transformation that hides elements of a vector, and elements and structure of a matrix, which can be employed to encrypt a GP problem.

2.3.1 Privacy-Preserving Vector Addition

The client can efficiently hide a private variable vector by adding a randomly generated vector to it. Specifically, a private variable vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)^t$ can be encrypted as follows:

$$\mathbf{y} = \mathbf{x} + \mathbf{r} \tag{2.4}$$

where $y_i = x_i + r_i$ for any $i \in [1, n]$, and $y_i, x_i,$ and r_i are the ith element of vector $\mathbf{y}, \mathbf{x}$ and $\mathbf{r}$, respectively. We assume that x_i is within the range $[-K, K]$, where $K = 2^l (l > 0)$ is a positive constant. Additionally, vector $\mathbf{r} \in R^{n \times 1}$ is randomly generated with its elements subject to uniform distribution and the corresponding probability density function is expressed as:

$$f(r_i) = \begin{cases} \dfrac{1}{2c} & -c \leq r_i \leq c \\ 0 & \text{otherwise} \end{cases} \tag{2.5}$$

where $c = 2^{l+p} (p > 0)$ is a positive constant, and $r_i, i \in [1, n]$ is the ith element of vector $\mathbf{r}$. Next we will obtain the following theorem that vectors $\mathbf{r}$ and $\mathbf{y}$ are computationally indistinguishable.

Theorem 1. Let $\mathbf{r}$ be a random vector with elements sampled from a uniform distribution with interval $[-c, c]$. Then vectors $\mathbf{r}$ and $\mathbf{y} = \mathbf{x} + \mathbf{r}$ are computationally indistinguishable.

Proof. According to Definition 1, we need to prove that any probabilistic polynomial time distinguisher D cannot distinguish y_i from r_i for any $i \in [1, n]$ except with negligible success probability, where y_i and r_i are the ith element of vector $\mathbf{y}$ and $\mathbf{r}$ respectively. The best strategy for a polynomial time distinguisher D when presented with a sample y_i is to return $b \leftarrow \{0, 1\}$ with equal probability if $-c \leq y_i \leq c$, and 1 if $y_i < -c$ or $y_i > c$. Therefore, the success probability of the distinguisher with the input being $y_i = x_i + r_i$ is given by

$$\begin{aligned} Pr[D(y_i) = 1] =& \frac{1}{2} Pr[-c \leq x_i + r_i \leq c] \\ & + Pr[x_i + r_i < -c] + Pr[x_i + r_i > c] \\ =& \frac{1}{2}(1 - Pr[x_i + r_i < -c] - Pr[x_i + r_i > c]) \\ & + Pr[x_i + r_i < -c] + Pr[x_i + r_i > c] \end{aligned} \tag{2.6}$$

Recall that x_i is within the range $[-K, K]$, and r_i is sampled from a uniform distribution specified by (2.5). We have that

$$\begin{aligned} Pr[x_i + r_i > c] &= Pr[r_i > c - x_i] \\ &\leq Pr[r_i > c - K] = \frac{K}{2c} \end{aligned} \tag{2.7}$$

Similarly, we find that $Pr[x_i + r_i < -c] \leq \frac{K}{2c}$. Consequently, we have that the success probability of the distinguisher D is bounded as follows:

$$Pr[D(y_i) = 1] \leq \frac{1}{2} + \frac{K}{2c} \tag{2.8}$$

On the other hand, when the input is r_i, obviously we can obtain that:

$$Pr[D(r_i) = 1] = \frac{1}{2} \tag{2.9}$$

According to Eq. (2), for any $i \in [1, n]$, we get that

$$|Pr[D(y_i) = 1] - Pr[D(r_i) = 1]| \leq \frac{K}{2c} \tag{2.10}$$

Note that $K = 2^l$ and $c = 2^{l+p}$. Hence, we have

$$\mu(p) = \frac{K}{2c} \leq \frac{2^l}{2^{l+p+1}} = \frac{1}{2^{p+1}} \tag{2.11}$$

which is a negligible function for large p. This concludes the proof. □

2.3.2 Privacy-Preserving Matrix Multiplication

The client efficiently encrypts the values of its private problem matrix by performing sparse random matrix multiplications. In particular, a private matrix $\mathbf{H} \in R^{m \times n}$ can be efficiently encrypted by performing the following multiplications:

$$\tilde{\mathbf{H}} = \mathbf{DHF} \tag{2.12}$$

Here $\mathbf{D} \in R^{m \times m}$ is a diagonal matrix defined as

$$d_{i,j} = \begin{cases} v_i & i = j \text{ for } i, j \in [1, m] \\ 0 & \text{otherwise} \end{cases} \tag{2.13}$$

where the value v_i, for $i \in [1, m]$, is generated based on the uniform distribution defined in (2.5). $\mathbf{F} \in R^{n \times n}$ is also a diagonal matrix with elements being arbitrary positive constant M. Consequently, the elements of $\tilde{\mathbf{H}}$ in (2.12) are given by

$$\tilde{h}_{i,j} = d_{i,i} h_{i,j} f_{j,j} = v_i h_{i,j} M \tag{2.14}$$

Assume that the element values of matrix $\mathbf{H}$ are within the range $[-T, T]$. Next we can arrive at Theorem 2 that the encrypted private matrix $\tilde{\mathbf{H}}$ and a random matrix $\mathbf{R}$ with elements sampled from a uniform distribution are computationally indistinguishable.

Theorem 2. Let $\mathbf{R} \in R^{m \times n}$ be a random matrix with elements in its jth column sampled from a uniform distribution with interval $[-c, c]$, for $j \in [1, n]$. Matrices $\mathbf{R}$ and $\tilde{\mathbf{H}}$ are computationally indistinguishable.

Proof. According to Definition 2, we need to prove that $r_{i,j}$ and $\tilde{h}_{i,j}$, for $i \in [1, m], j \in [1, n]$, are computationally indistinguishable for matrices $\mathbf{R}$ and $\tilde{\mathbf{H}}$ to be computationally indistinguishable. Specifically, we prove that any polynomial time distinguisher D cannot distinguish $\tilde{h}_{i,j}$ from $r_{i,j}$, for $i \in [1, m], j \in [1, n]$, except with negligible success probability.

The distinguisher D is defined in the same way as in Theorem 1. Therefore, the

success probability of the distinguisher D is given by

$$\begin{aligned} Pr[D(\tilde{h}_{i,j}) = 1] =& \frac{1}{2} Pr[-c \leq \tilde{h}_{i,j} \leq c] \\ & + Pr[\tilde{h}_{i,j} < -c] + Pr[\tilde{h}_{i,j} > c] \\ =& \frac{1}{2}(1 - Pr[\tilde{h}_{i,j} < -c] - Pr[\tilde{h}_{i,j} > c]) \\ & + Pr[\tilde{h}_{i,j} < -c] + Pr[\tilde{h}_{i,j} > c] \end{aligned} \tag{2.15}$$

where

$$\begin{aligned} Pr[\tilde{h}_{i,j} > c] &= Pr[v_i h_{i,j} M > c] \\ &= Pr[v_i h_{i,j} > \frac{c}{M}] \\ &\leq \gamma Pr[v_i > \frac{c}{MT}] + (1-\gamma) Pr[v_i < \frac{-c}{MT}] \\ &= \frac{1}{2} - \frac{1}{2MT} \end{aligned} \tag{2.16}$$

the parameter γ is the probability of the element $h_{i,j}$ being positive and $1-\gamma$ is $h_{i,j}$ being negative. Similarly, we find that $Pr[\tilde{h}_{i,j} < -c] \leq \frac{1}{2} - \frac{1}{2MT}$. Consequently, we have that the success probability of distinguisher D is bounded as follows:

$$Pr[D(\tilde{h}_{i,j}) = 1] = 1 - \frac{1}{2MT} \tag{2.17}$$

On the other hand, we can easily obtain that $Pr[D(r_{i,j}) = 1] = \frac{1}{2}$.

According to Eq. (2.3), for $i \in [1, m], j \in [1, n]$, it follows that

$$|Pr[D(\tilde{h}_{i,j}) = 1] - Pr[D(r_{i,j}) = 1]| \leq \frac{MT - 1}{2MT} \tag{2.18}$$

Note that M is an arbitrary positive constant, we have

$$\beta(M) = \frac{MT - 1}{2MT} \tag{2.19}$$

Thus, $\beta(M)$ can be guaranteed as a negligible function when MT approaches 1. This concludes the proof. □

2.3.3 Privacy-Preserving Matrix Permutation

Although the matrix transformation in Eq. (2.12) hides the values of the elements in $\mathbf{H}$, it still reveals the original positions of the non-zero elements, i.e., $\mathbf{H}$'s structure, which is also private. Next, we design secure permutations that can hide $\mathbf{H}$'s structure by randomly reordering the rows and columns of $\mathbf{H}$.

The client applies the random permutations as follows:

$$\hat{\mathbf{H}} = \mathbf{E}\tilde{\mathbf{H}}\mathbf{U} \tag{2.20}$$

where $\mathbf{E} \in R^{m\times m}$ and $\mathbf{U} \in R^{n\times n}$ are random permutation matrices, and their elements are defined by

$$\begin{aligned} e_{i,j} &= \delta_{\pi(i),j} \ \forall i \in [1,m], j \in [1,m] \\ u_{i,j} &= \delta_{\pi(i),j} \ \forall i \in [1,n], j \in [1,n] \end{aligned} \tag{2.21}$$

where i and j are the row and column indexes, respectively. The random permutation function $\pi(\cdot)$ maps an original index $i \in \{1, 2, \cdots, n\}$ to its permuted index within the same range. Besides, the Kronecker delta function as defined in [11] is given by

$$\delta_{i,j} = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases} \tag{2.22}$$

The details of generating random permutation matrices in Eq. (2.20) are summarized in Algorithm 1.

In addition, the client is able to recover the original matrix $\tilde{\mathbf{H}}$ by applying the following inverse permutations:

Algorithm 1 Random permutation matrix generation

Input: Initial index set $N = \{1, 2, \ldots, n\}$
Output: Random permutation matrix $\mathbf{P}$
Set $\pi = I_n$; (identical permutation)
for $i = n$ down to 2 **do**
  Set j to a random integer with $1 \leq j \leq i$;
  Swap $\pi[i]$ and $\pi[j]$ in set N;
end for
for $i = 1$ to n **do**
  for $j = 1$ to n **do**
    $\pi(i)$ outputs the ith element in set N;
    $\delta_{\pi(i),j}$ outputs value based on Eq. (2.22);
    Set $\mathbf{P}(i, j) = \delta_{\pi(i),j}$;
  end for
end for
return $\mathbf{P}$;

$$\tilde{\mathbf{H}} = \mathbf{E}^{\mathbf{T}}\hat{\mathbf{H}}\mathbf{U}^{\mathbf{T}} \tag{2.23}$$

To get this result, the orthogonal property of permutation matrices are applied, i.e., $\mathbf{E}^{\mathbf{T}}\mathbf{E} = \mathbf{I}$ and $\mathbf{U}^{\mathbf{T}}\mathbf{U} = \mathbf{I}$, where $\mathbf{I}$ is the identity matrix.

2.4 The Lagrange Dual Problem

Since the objective function f_0 and constraint functions g_k in GP problem (2.1) are posynomials and are non-convex in general [12], GP problem (2.1) is a non-convex optimization problem and it can take exponential time to solve this problem, especially with large-scale decision variables and constraints. In this section, we identify an equivalent dual optimization problem that is convex with only linear constraints, i.e., the dual geometric problem.

In particular, first we transform the original non-convex GP problem with the following variable substitution:

$$\mathbf{y} = \log \mathbf{x} \tag{2.24}$$

where $\mathbf{y} = (y_1, y_2, \ldots, y_n) \in R^{n\times 1}, \mathbf{x} = (x_1, x_2, \ldots, x_n) \in R^{n\times 1}$.

Next, from (2.24) we denote

$$\tau_j = c_j \prod_{i=1}^{n} x_i^{a_{ij}} = c_j e^{\mathbf{a_j^t y}} \text{ for } j = 1, \ldots, N_m \tag{2.25}$$

where $\mathbf{a_j} = (a_{j1}, \ldots, a_{jn})^t$ for $j = 1, \ldots, N_m$. Taking a logarithmic transformation of the objective and constraint functions, the original GP problem (2.1) can be equivalently rewritten as:

$$\begin{aligned} &\text{Minimize} && \log[F(\mathbf{y})] \\ &\text{subject to} && \log[G_k(\mathbf{y})] \leq 0,\ k = 1, \ldots, m. \\ & && \mathbf{y} \text{ unrestricted in sign} \end{aligned} \tag{2.26}$$

where

$$\begin{aligned} F(\mathbf{y}) &= \sum_{j=1}^{N_0} \tau_j \\ G_k(\mathbf{y}) &= \sum_{j=N_{k-1}+1}^{N_k} \tau_j \text{ for } k = 1, \ldots, m \end{aligned} \tag{2.27}$$

According to [13], the problem (2.26) is now a convex programming problem.

In the following, we use the Lagrangian dual approach to solve problem (2.26). Since the interiority constraint qualification holds, there is no gap between problem (2.26) and its Lagrangian dual stated below:

$$\begin{aligned} \boldsymbol{LD}: &\text{ Maximize } L(\mathbf{y}, \mathbf{u}) \\ &\boldsymbol{\nabla}_{\mathbf{y}} L(\mathbf{y}, \mathbf{u}) = \mathbf{0} \\ &\mathbf{u} \geq \mathbf{0}, \mathbf{y} \text{ unrestricted} \end{aligned} \tag{2.28}$$

where the Lagrangian function is

$$L(\mathbf{y}, \mathbf{u}) = \log[F(\mathbf{y})] + \sum_{i=1}^{m} u_i \log[G_i(\mathbf{y})] \tag{2.29}$$

We now define a new variable vector $\boldsymbol{\delta}$ as follows:

$$\begin{aligned} \delta_k &= \frac{\tau_k}{F} \text{ for all } k \in [1, N_0] \\ \delta_k &= \frac{u_i \tau_k}{G_i} \text{ for all } k \in [N_0 + 1, N_m] \end{aligned} \tag{2.30}$$

Note that, from Eq. (2.27) and (2.30), we have

$$\begin{aligned} \sum_{k=1}^{N_0} \delta_k &= 1 \\ \sum_{k=N_0+1}^{N_m} \delta_k &= u_i \text{ for } i \in [1, m] \end{aligned} \tag{2.31}$$

According to $\boldsymbol{\nabla}_{\mathbf{y}} L(\mathbf{y}, \mathbf{u}) = \mathbf{0}$ in the ***LD*** problem (2.28) and Eq. (2.30), we have

$$\begin{aligned} \boldsymbol{\nabla}_{\mathbf{y}} L(\mathbf{y}, \mathbf{u}) &= \frac{\boldsymbol{\nabla} F(\mathbf{y})}{F(\mathbf{y}} + \sum_{i=1}^{m} u_i \frac{\boldsymbol{\nabla} G_i(\mathbf{y})}{G_i(\mathbf{y})} \\ &= \frac{1}{F(\mathbf{y})} \sum_{k=1}^{N_0} \tau_k \boldsymbol{a}_k + \sum_{i=1}^{m} \frac{u_i}{G_i(\mathbf{y})} \left[\sum_{k=N_0+1}^{N_m} \tau_k \boldsymbol{a}_k \right] \\ &= \sum_{k=1}^{N_m} \delta_k \boldsymbol{a}_k = \mathbf{0} \end{aligned} \tag{2.32}$$

From Eq. (2.25), (2.30), and (2.31), the term $u_i \log[G_i(\mathbf{y})]$ in Eq. (2.29) can also be

rewritten as:

$$\begin{aligned} u_i \log[G_i(\mathbf{y})] &= u_i \log(u_i) + u_i \log\left[\frac{G_i}{u_i}\right] \\ &= u_i \log(u_i) + \sum_{k=N_0+1}^{N_m} \delta_k \log\left[\frac{\tau_k}{\delta_k}\right] \\ &= u_i \log(u_i) + \sum_{k=N_0+1}^{N_m} \delta_k \log\left[\frac{c_k}{\delta_k} e^{\boldsymbol{a_k y}}\right] \\ &= u_i \log(u_i) + \sum_{k=N_0+1}^{N_m} \delta_k \log\left[\frac{c_k}{\delta_k}\right] + \sum_{k=N_0+1}^{N_m} \delta_k \boldsymbol{a_k y} \end{aligned} \tag{2.33}$$

Similarly, we have

$$\log[F(\mathbf{y})] = \sum_{k=0}^{N_0} \delta_k \log\left[\frac{c_k}{\delta_k}\right] + \sum_{k=0}^{N_0} \delta_k \boldsymbol{a_k y} \tag{2.34}$$

Thus, from Eq. (2.33) and (2.34), the objective function (2.29) can be rewritten as:

$$L(\boldsymbol{\delta}, \mathbf{u}) = \sum_{k=1}^{N_m} \delta_k \log\left[\frac{c_k}{\delta_k}\right] + \sum_{i=1}^{m} u_i \log(u_i) \tag{2.35}$$

We finally use Eq. (2.31), (2.32), and (2.35) to replace the ***LD*** problem (2.28) with

the following *dual geometric program (DGP)* in the variables $(\boldsymbol{\delta}, \mathbf{u})$:

$$
\begin{aligned}
\boldsymbol{DGP}: \text{Maximize} \quad & \sum_{k=1}^{N_m} \delta_k \log\left[\frac{c_k}{\delta_k}\right] + \sum_{i=1}^{m} u_i \log(u_i) \\
\text{subject to} \quad & \mathbf{A}\boldsymbol{\delta} = \mathbf{0} \\
& \sum_{k=0}^{N_0} \delta_k = 1 \\
& \sum_{k=N_0+1}^{N_m} \delta_k - u_i = 0 \text{ for } i \in [1, m] \\
& \boldsymbol{\delta} \geq \mathbf{0} \\
& \boldsymbol{u} \geq \mathbf{0}
\end{aligned}
\tag{2.36}
$$

where $\mathbf{A} = [\mathbf{a_1}\ \mathbf{a_2}\ \ldots\ \mathbf{a_k}] \in R^{n \times N_m}$.

Note that the *DGP* problem (2.36) is a convex programming problem with linear constraints. We denote the optimal solution to (2.36) as $(\boldsymbol{\delta^*}, \boldsymbol{u^*})$.

Since the DGP problem (2.36) is convex and the affine constraints are feasible, the strong duality holds [13] and according to Eq. (2.24), (2.25), and (2.30), we have that

$$
\begin{aligned}
\mathbf{y}^* &= (\mathbf{A^T})^{-1} \log(\boldsymbol{\delta}^*/\mathbf{u}^*\mathbf{c}) \\
\mathbf{x}^* &= \mathbf{e}^{\mathbf{y}^*}
\end{aligned}
\tag{2.37}
$$

where $\mathbf{c} = (c_1, c_2, \ldots, c_{N_m})^t$. That is, we can use the result of the DGP problem (2.36) to recover the result of the original GP problem (2.1).

2.5 Solving the Outsourced Problem

In this section, we describe a secure and efficient algorithm to solve the large-scale GP problems based on the DGP problem derived in Section 2.4.

2.5.1 An Iterative Solution

Before we delve into details about the proposed algorithm, we first present the gradient projection method (GPM), an iterative solution method for nonlinear convex optimization problems that we employ to solve the DGP problem in (2.36).

For optimization problems, the search direction of fastest descent is the negative gradient of the objective function. However, moving along the negative gradient may lead to violating the constraint functions. The main idea of the GPM is to project the negative gradient in such a way that improves the objective function while not violating any constraints.

In particular, let's first consider the following convex optimization problem

$$\begin{aligned} \text{Minimize} \quad & f(\mathbf{z}) \\ \text{subject to} \quad & \mathbf{A}'\mathbf{z} = \mathbf{b}' \\ & \mathbf{z} \geq \mathbf{0} \end{aligned} \tag{2.38}$$

Given a feasible point $\mathbf{z}$, the moving direction of steep descent is $-\nabla f(\mathbf{z})$. However, moving along $-\nabla f(\mathbf{z})$ may violate feasibility of the solutions. To this end, the moving direction $\mathbf{d}$ is projected so that $\mathbf{d} = -\mathbf{P}\nabla\mathbf{f}(\mathbf{z})$, where $\mathbf{P}$ is a suitable projection matrix [13]. The following Theorem 3 [13, 14] will provide a way to find an improving feasible direction $\mathbf{d}$.

Theorem 3. Consider the optimization problem in (2.38), and suppose $f(\mathbf{z})$ is differentiable at the point $\mathbf{z}$. Let projection matrix $\mathbf{P}$ be of the form $\mathbf{P} = \mathbf{I} - \mathbf{A'^T}(\mathbf{A'A'^T})^{-1}\mathbf{A}$. Then $\mathbf{d} = -\mathbf{P}\nabla\mathbf{f}(\mathbf{z})$ is an improving feasible direction if $\mathbf{d} \neq \mathbf{0}$, and $\mathbf{z}$ is a KKT point if $\mathbf{d} = \mathbf{0}$.

Proof. First of all, it is noted that

$$\begin{aligned}\mathbf{A'd} &= -\mathbf{A'P}\nabla\mathbf{f(z)} \\ &= -\mathbf{A'}(\mathbf{I} - \mathbf{A'^T}(\mathbf{A'A'^T})^{-1}\mathbf{A'})\nabla\mathbf{f(z)} \\ &= \mathbf{0}\end{aligned} \tag{2.39}$$

Thus $\mathbf{d}$ is a feasible direction. In addition, according to the properties of projection matrix , i.e., $\mathbf{P} = \mathbf{P^T},\ \mathbf{P} = \mathbf{P^2}$ [13], we have

$$\begin{aligned}\nabla\mathbf{f(z)^T d} &= -\nabla\mathbf{f(z)^T P}\nabla\mathbf{f(z)} \\ &= -\nabla\mathbf{f(z)^T P^T P}\nabla\mathbf{f(z)} \\ &= -||\mathbf{P}\nabla\mathbf{f(z)}||^{\mathbf{2}} < 0\end{aligned} \tag{2.40}$$

Thus $\mathbf{d}$ is also an improving direction if $\mathbf{d} \neq \mathbf{0}$. From Eq. (3.8), (3.9), we have $\mathbf{d}$ is an improving feasible direction, which completes the proof. □

Once an improving feasible direction $\mathbf{d}$ is found, the optimal point of the objective function $f(\mathbf{z})$ can be approached in the following iterative way:

$$\mathbf{z}_{k+1} = \mathbf{z}_k + \lambda_k \mathbf{d}_k \tag{2.41}$$

where k is the iteration index, and the upper bound of λ_k defined as λ_{max} is given by:

$$\lambda_{max} = \begin{cases} \min_{1\leq i\leq n} : \left\{ \dfrac{-z_{ik}}{d_{ik}} : d_{ik} < 0 \right\} & \mathbf{d_k} \ngeq \mathbf{0} \\ \infty & \mathbf{d_k} \geq \mathbf{0} \end{cases} \tag{2.42}$$

where z_{ik} and d_{ik} are the ith elements of $\mathbf{z}_k$ and $\mathbf{d}_k$, respectively. Hence, the value of λ_k is determined by the following line search problem:

$$\begin{aligned} &\text{Minimize} \quad f(\mathbf{z}_k + \lambda \mathbf{d}_k) \\ &\text{subject to} \quad 0 \leq \lambda \leq \lambda_{max} \end{aligned} \tag{2.43}$$

Next, since problem (2.38) consists of a convex objective function with only linear constraints, the GPM converges. The proof for convergence of the GPM is omitted here due to the space limitation.

2.5.2 A Secure Algorithm For Solving Large-scale GP

As shown in Section 2.4, the client transforms the original GP problem (2.1) to the DGP problem (2.36), and the DGP problem is outsourced to the cloud server for solving. To protect the client's data privacy, we conduct some transformations based on the proposed scheme in Section 2.3.

For convenience, we first rewrite the DGP problem in (2.36) as the following form:

$$\begin{aligned} \text{Minimize} \quad & D(\mathbf{z}) \\ \text{subject to} \quad & \mathbf{W}\mathbf{z} = \mathbf{b} \\ & \mathbf{z} \geq \mathbf{0} \end{aligned} \tag{2.44}$$

where

$$D(\mathbf{z}) = -L(\boldsymbol{\delta}, \mathbf{u}) = -\sum_{k=1}^{N_m} \delta_k \log\left[\frac{c_k}{\delta_k}\right] - \sum_{i=1}^{m} u_i \log(u_i)$$

$$\mathbf{b} = (0, 1, 0, \ldots, 0)^t \in R^{(n+m+1)\times 1}$$

$$\mathbf{z} = (\boldsymbol{\delta}, \mathbf{u})^t \in R^{(N_m+m)\times 1}$$

Thus, to protect the sensitive information of the coefficient matrix, the client applies the matrix transformations (2.12) and (2.20) to $\mathbf{W}$ in (2.44) as follows:

$$\bar{\mathbf{W}} = \mathbf{V}\mathbf{W}\mathbf{T} \tag{2.45}$$

where $\mathbf{V}$ is formed by a random permutation matrix and a random diagonal matrix, i.e., $\mathbf{V} = \mathbf{E}\mathbf{D}$, and $\mathbf{T}$ formed by a diagonal matrix of arbitrary positive constant and a random permutation matrix, i.e., $\mathbf{T} = \mathbf{F}\mathbf{U}$.

Furthermore, to protect the privacy of decision variable vector $\mathbf{z}$, the vector transformation (2.4) is applied in the following:

$$\bar{\mathbf{z}} = \mathbf{T}^{-\mathbf{1}}(\mathbf{z} + \mathbf{r}) \tag{2.46}$$

where $\mathbf{r} \in R^{(N_m+m)\times 1}$ is a random vector. Based on Eq. (2.45) and (2.46) we have

$$\bar{\mathbf{b}} = \bar{\mathbf{W}}\bar{\mathbf{z}} = \mathbf{V}(\mathbf{b} + \mathbf{W}\mathbf{r}) \tag{2.47}$$

Now we can transform the problem (2.44) into the following privacy-preserving problem:

$$\begin{aligned} \text{Minimize} \quad & D(\bar{\mathbf{z}}) \\ \text{subject to} \quad & \bar{\mathbf{W}}\bar{\mathbf{z}} = \bar{\mathbf{b}} \\ & \bar{\mathbf{z}} \geq \mathbf{T}^{-\mathbf{1}}\mathbf{r} \end{aligned} \tag{2.48}$$

Next, the encrypted problem (2.48) will be sent to the cloud server, and the GPM will be applied to solve it. The complete procedure is summarized in Algorithm 2.

Algorithm 2 Secure algorithm for solving outsourced large-scale GP

Input: Starting point $\mathbf{z}_0$ that $\bar{\mathbf{W}}\bar{\mathbf{z}}_0 = \bar{\mathbf{b}}$
Output: Optimal point $\bar{\mathbf{z}}^*$ for problem (2.48)

Initialize $k = 0$;
Compute $\mathbf{P}$ and $\mathbf{d}_0$ from Theorem 3;
Let $\mathbf{d} = \mathbf{d}_0$;
while $\mathbf{d} \neq \mathbf{0}$ **do**
    Compute λ_{max} using Eq. (2.42);
    Solve the line search for λ_k:
    $\lambda_k = \text{argmin}(0 \leq x \leq \lambda_{max})\{\ \text{f}(\text{z}_k + \lambda \mathbf{d}_k)\}$;
    Let $\bar{\mathbf{z}}_{k+1} = \bar{\mathbf{z}}_k + \lambda_k \mathbf{d}_k$;
    $k = k + 1$;
    Compute $\mathbf{d}_k$ from Theorem 3;
    Let $\mathbf{d} = \mathbf{d}_k$;
end while
Let $\bar{\mathbf{z}}^* = \bar{\mathbf{z}}_{\mathbf{k}}$, and $\bar{\mathbf{z}}^*$ is a KKT point;
return $\bar{\mathbf{z}}^*$;

As stated in Algorithm 2, the cloud server continues the iteration until the secure outsourcing algorithm converges to the KKT point $\bar{\mathbf{z}}^*$. Once the cloud server determines that the algorithm has converged, it sends $\bar{\mathbf{z}}^*$ back to the client, who verifies the correctness of the returned result based on KKT conditions. If the returned result satisfies the KKT conditions, the client determines the returned result is correct and compute the solution to the DGP problem (2.36) as follows:

$$\mathbf{z}^* = \mathbf{T}\bar{\mathbf{z}}^* - \mathbf{r} \tag{2.49}$$

From (2.44), we have $\mathbf{z}^* = (\boldsymbol{\delta}^*, \mathbf{u}^*)^{\mathbf{t}}$. Thus the solution to the original GP problem (2.1) can be obtained by following Eq. (2.37).

2.6 Performance Evaluation

This section presents the computational complexity of the proposed solution and experiment results.

2.6.1 Computational Complexity

The computational cost at each step is analyzed as follows. First, the client transforms the original GP problem to the DGP problem, which takes $\mathcal{O}(m^2)$ computational cost. Then, the client employs the privacy-preserving matrix transformation to encrypt the coefficient matrices, which induces a computational complexity of $\mathcal{O}(2m^2 + 4mn)$. Next, the cloud server solves the outsourced problem, which needs computation of $\mathcal{O}(2m^2n + m^3)$. Lastly, the client recovers the optimal solution based on the returned solution from the cloud server, resulting in a computational complexity of $\mathcal{O}(2mn)$.

To summarize, the proposed privacy-preserving outsourcing protocol requires computational complexity of $\mathcal{O}(\max\{mn, m^2\})$ at the client side and that of $\mathcal{O}(\max\{m^2n, m^3\})$ at the cloud side.

Table 2.1: Computing Time (12 cloud nodes, 16GB memory per node)

# of Variables	Solving GP by Client	Client's Computing in Our Solution	Cloud's Computing in Our Solution
1,000	21.6s	0.31s	6.7s
2,000	43.9s	0.39s	13.1s
3,000	77.6s	0.52s	29.2s
4,000	137.7s	0.65s	48.1s
5,000	254.9s	0.86s	79.6s
6,000	355.7s	1.03s	98.4s
7,000	536.1s	1.34s	142.7s
8,000	1352.7s	2.17s	301.8s

2.6.2 Experiment Results

In the following, we evaluate the performance of the proposed privacy-preserving outsourcing protocol for GP through experiments. We implemented the proposed protocol in a real-world scenario. The client side was implemented on a laptop with a dual-core 2.3 GHz CPU, 8GB RAM, and 256 GB solid state drive. The cloud side was implemented on the Amazon Elastic Compute Cloud (EC2) with a number of computing nodes each of 16GB memory. Both the client-side and the cloud-side computations were implemented by Matlab R2018a. The GP problems used in evaluations are randomly generated. Each data point presented below is the average of 20 runs with different randomness seeds.

We first measure the computing time of the proposed protocol at both the client and the cloud side. In these experiments, we only used 12 cloud nodes. Table 2.1 shows the results. It can be observed that the client can complete the needed computations very quickly, even for large-size GPs. For example, it only takes the client 2.2s to complete the computation for GP problem size 8000 (i.e., parameter m). The computing time of the cloud server to obtain the optimal solution is much longer than the client due to the complex nature of solving the problem. Not surprisingly, the computation time at both the client side and the cloud side increases when the problem size increases.

Subsequently, we examine the computing saved for the client by our outsourcing

protocol. As shown in Table 2.1, we compare the computing time of the client when it solves GP by itself with that when it outsources GP to the cloud. The saved computing increases dramatically as the problem size increases. For example, the saving can reach 624-fold for problem size 8000, indicating a 99.8% reduction in computing at the client. This validates the efficacy of our proposed protocol for the client.

The overall delay for solving GP is also much shorter when outsourcing it to the more powerful cloud. When the client solves GP by itself, the delay is the time shown in the second column of Table 2.1. When the client outsources GP to the cloud using our protocol, the overall delay is the client's computing time (the 3rd column of Table 2.1) plus the cloud's computing time (the 4th column of Table 2.1). Here when delay is concerned the communication time between the client and the cloud is neglected since it is much shorter than the computing time. Then from Table 2.1, it can be seen that the overall problem resolving delay is several times shorter in our solution.

Next, we investigate the computing time at the cloud server when a varying number of nodes are used. The results are shown in Fig. 2.2. It can be observed that the computing time of the cloud server decreases as the number of nodes used grows. For example, the computing time is as low as 302s when 12 nodes are used compared with about 520s when only 4 nodes are used. The computing time of the cloud server can be further shortened by using more cloud nodes.

Lastly, we measure the computing time of the cloud server with different node memory sizes. As it can be seen from Fig. 2.3, the computing time at the cloud server decreases as the node memory increases. For example, when node memory size increases from 8GB to 32GB, the computing time decreases from 372s to 287s for problem size 8000, indicating a huge time saving at the cloud side. We also noticed that when the memory size increases from 16GB to 32GB, the reduction in computing time is only a little compared with the

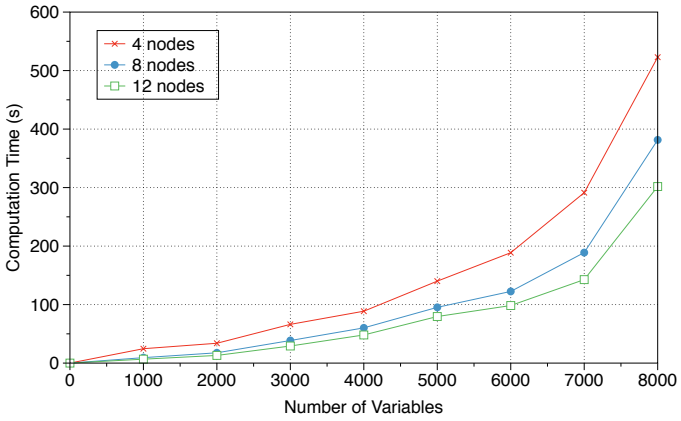


Figure 2.2: Computing time of cloud server with different node sizes

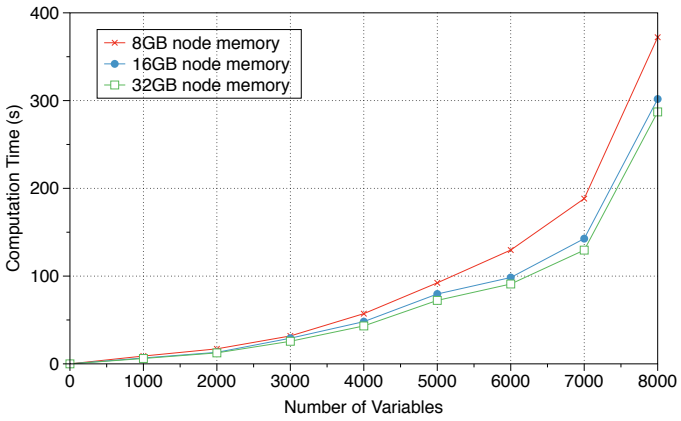


Figure 2.3: Computing time of cloud server with different node memory sizes

reduction when the memory size increases from 8GB to 16GB. That is because 16GB node memory is already good for the maximum problem size experimented, i.e., 8000. When the problem size is larger, the reduction in computing time when node memory increases from 16GB to 32GB should be more.

2.7 Related Work

In recent years, researchers have developed many protocols for privacy-preserving cloud computing.

Based on fully homomorphic encryption (FHE) [15], Gennaro et al. [16] proposed a privacy-preserving outsourcing algorithm by employing fully homomorphic encryption

(FHE). Wang et al. [17] developed an iterative algorithm to solve linear systems of equations, where a client transforms and encrypts the coefficient matrix using homomorphic encryption, and the cloud carries out computations on ciphertexts. Hu et al. [18] designed secure interactive protocols to distribute the feature extraction computations to two independent cloud servers. However, these algorithms require the client to perform extensive data pre-processing and encryption/decryption operations.

Without resorting to homomorphic encryption, Wang et al. [19] presented an efficient algorithm to securely compute histogram of oriented gradients based on matrix transformations. Du et al. [20] designed a secure outsourcing protocol for the non-linear programming problem by applying the reduced gradient method. Shen et al. [21] developed a secure outsourcing scheme to solve linear algebraic equations. Some secure outsourcing protocols for matrix computation have also been developed, such as matrix inversion [22], matrix multiplication [23], and matrix determinant [24]. Besides, Zhang et al. [25] considered employing matrix digest techniques to securely outsource batch matrix multiplication. However, the privacy-preserving outsourcing algorithm for large-scale GPs has not been studied so far.

2.8 Summary

In this chapter, we investigated privacy-preserving outsourcing of large-scale geometric programming problems. To the best of our knowledge, this is the first work to solve geometric programming in cloud computing with privacy protection. We employed a transformation scheme to protect the client's private data, and formally proved its effectiveness. The gradient projection method was used by the cloud server to solve the transformed geometric programming problem. Experimental results based on Amazon EC2 showed that the proposed protocol can provide significant time savings to the client.

Bibliography

[1] E. Aarts and J. Korst, "Simulated annealing and boltzmann machines," 1988.

[2] D. P. Palomar and Y. C. Eldar, *Convex optimization in signal processing and communications*. Cambridge university press, 2010.

[3] S. A. Zenios, *Financial optimization*. Cambridge university press, 2002.

[4] P. Khosravi, Y. Liang, Y. Choi, and G. V. d. Broeck, "What to expect of classifiers? reasoning about logistic regression with missing features," *arXiv preprint arXiv:1903.01620*, 2019.

[5] S. Fu, Q. Fan, Y. Tang, H. Zhang, X. Jian, and X. Zeng, "Cooperative computing in integrated blockchain-based internet of things," *IEEE Internet of Things Journal*, vol. 7, no. 3, pp. 1603–1612, 2019.

[6] S. Yu, C. Wang, K. Ren, and W. Lou, "Achieving secure, scalable, and fine-grained data access control in cloud computing," in *Infocom, 2010 proceedings IEEE*, pp. 1–9, Ieee, 2010.

[7] F. Chen, T. Xiang, and Y. Yang, "Privacy-preserving and verifiable protocols for scientific computation outsourcing to the cloud," *Journal of Parallel and Distributed Computing*, vol. 74, no. 3, pp. 2141–2151, 2014.

[8] W. Bao and Q. Li, "Efficient privacy-preserving outsourcing of large-scale geometric programming," in *2018 IEEE Symposium on Privacy-Aware Computing (PAC)*, pp. 55–63, IEEE, 2018.

[9] S. Boyd, S.-J. Kim, L. Vandenberghe, and A. Hassibi, "A tutorial on geometric programming," *Optimization and engineering*, vol. 8, no. 1, p. 67, 2007.

[10] J. Katz and Y. Lindell, *Introduction to modern cryptography*. CRC press, 2014.

[11] L. Zhou and C. Li, "Outsourcing large-scale quadratic programming to a public cloud," *IEEE Access*, vol. 3, pp. 2581–2589, 2015.

[12] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[13] D. P. Bertsekas, *Nonlinear programming*. Athena scientific Belmont, 1999.

[14] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty, *Nonlinear programming: theory and algorithms*. John Wiley & Sons, 2013.

[15] C. Gentry *et al.*, "Fully homomorphic encryption using ideal lattices.," in *STOC*, vol. 9, pp. 169–178, 2009.

[16] R. Gennaro, C. Gentry, and B. Parno, "Non-interactive verifiable computing: Outsourcing computation to untrusted workers," in *Annual Cryptology Conference*, pp. 465–482, Springer, 2010.

[17] C. Wang, K. Ren, J. Wang, and Q. Wang, "Harnessing the cloud for securely outsourcing large-scale systems of linear equations," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 6, pp. 1172–1181, 2013.

[18] S. Hu, Q. Wang, J. Wang, Z. Qin, and K. Ren, "Securing sift: Privacy-preserving outsourcing computation of feature extractions over encrypted image data," *IEEE Transactions on Image Processing*, vol. 25, no. 7, pp. 3411–3425, 2016.

[19] Q. Wang, J. Wang, S. Hu, Q. Zou, and K. Ren, "Sechog: Privacy-preserving outsourcing computation of histogram of oriented gradients in the cloud," in *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*, pp. 257–268, ACM, 2016.

[20] W. Du and Q. Li, "Secure and efficient outsourcing of large-scale nonlinear programming," in *Communications and Network Security (CNS), 2017 IEEE Conference on*, pp. 1–9, IEEE, 2017.

[21] W. Shen, B. Yin, X. Cao, Y. Cheng, and X. S. Shen, "A distributed secure outsourcing scheme for solving linear algebraic equations in ad hoc clouds," *IEEE Transactions on Cloud Computing*, 2017.

[22] C. Hu, A. Alhothaily, A. Alrawais, X. Cheng, C. Sturtivant, and H. Liu, "A secure and verifiable outsourcing scheme for matrix inverse computation," in *INFOCOM 2017-IEEE Conference on Computer Communications, IEEE*, pp. 1–9, IEEE, 2017.

[23] X. Lei, X. Liao, T. Huang, and F. Heriniaina, "Achieving security, robust cheating resistance, and high-efficiency for outsourcing large matrix multiplication computation to a malicious cloud," *Information sciences*, vol. 280, pp. 205–217, 2014.

[24] S. Fu, Y. Yu, and M. Xu, "Practical privacy-preserving outsourcing of large-scale matrix determinant computation in the cloud," in *International Conference on Cloud Computing and Security*, pp. 3–15, Springer, 2017.

[25] X. Zhang, T. Jiang, K.-C. Li, A. Castiglione, and X. Chen, "New publicly verifiable computation for batch matrix multiplication," *Information Sciences*, 2017.

3 Privacy-Preserving Cloud-Assisted Distributed Logistic Regression

3.1 Introduction

In recent years, more and more data are generated in various domains. Industries and academia have been developing tools and techniques to transform data into useful knowledge aimed to improve various aspects of our life [1]. With the recent advance in computation power, machine learning algorithms have been widely adopted to analyze massive data and make accurate predictions due to their excellent performance in modeling complex patterns within data. Data sources, such as browsing histories in social media, smart wearables, and medical data, are usually distributed and could be used together to help improve machine learning algorithms.

While machine learning techniques are promising and gaining increasing popularity, there may exist private information in the training data used by the machine learning algorithms, and thus need to be protected. In the past decades, various privacy-preserving algorithms have been proposed. Motlagh et al [2] presented an association rule (AR) technique to protect user data. In terms of collaborative learning, secure multi-party computation (SMC) is proposed to protect intermediate results during training among distributed data owners. SMC has been used in various machine learning tasks, such as learning decision trees [3], linear regression functions [4], Naive Bayes classifiers [5], and k-means clustering [6]. However, those techniques are usually computation-extensive, making it impractical for large-scale applications.

Differential privacy [7] has also been widely adopted to protect user privacy. Theoretically, it could provide formal privacy guarantees no matter what extra information the attackers have. A lot of work have adopted differential privacy to protect user privacy in

the training process. A typical approach is first generating noise via Laplace mechanism or exponential mechanism and then building a noisy model for their dataset using these generated noises [8]. Some other approaches modify the objective function of the training model [9]. These mechanisms can perturb the objective function by adding noise to coefficients, and output predictions of the noisy model.

In fact, there exist various applications of logistic regression in academia and industry. However, only a few work have been proposed to utilize differential privacy in logistic regression algorithm. The main challenge would be that regression involves solving an optimization problem. It is usually difficult to analyze the relationship between the optimization results and the original data. Thus, it is hard to determine on the minimum amount of noise necessary to make the optimization results differentially private. In addition, some users may hold low-quality data possibly due to inaccurate data collection processes, which may impact the effectiveness of the trained model in distributed learning. The protection of privacy usually hides any accurate information of a user's data, making it difficult to check the data quality.

In this chapter, we propose a privacy-preserving distributed logistic regression framework that has high learning accuracy and is robust against low quality data of participating clients. In particular, first each client computes the magnitude of relevance [10] between the learning output and the input data features. Then, the local logistic regression learning objective is approximated with function of polynomial terms. Based on magnitudes of relevance for each data feature, different carefully-crafted noises are injected to the different coefficients of the polynomial objective function for achieving differential privacy; i.e. more noise is injected to the coefficients with less relevant features and vice-versa. Then, all the local parameters will be uploaded to a cloud server, which uses an evaluation dataset [11] to measure how good each client's data quality is. The cloud server selectively aggregates the

local parameters of a subset of clients based on their data quality and updates the global parameters in a way that preserves differential privacy. This process iterates until the training converges.

The main contributions of this chapter are summarized as follows:

- We propose a differentially private distributed logistic regression framework, which employs the relevance between input data features and the model output [10] to wisely add noise to the objective function and maintain good learning accuracy.
- Extensive experimental results show that the proposed framework can achieve low misclassification rate, robustness against low quality data, and strong privacy guarantee.

The chapter is organized as follows. Section 3.2 explains the preliminaries of logistic regression, differential privacy, and layer-wise relevance propagation. Section 3.3 presents our privacy-preserving algorithm for distributed logistic regression. Section 3.4 presents performance evaluation results. Section 3.5 discusses related work. Section 3.6 concludes the chapter.

3.2 Preliminaries

In this section, we provide a background introduction to logistic regression, differential privacy, and Layer-wise Relevance Propagation (LRP).

3.2.1 Logistic Regression

Logistic regression is a widely adopted machine learning algorithm and applied in different domains to solve regression and classification tasks. Given a database D with n feature and label tuples, i.e. $\{(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)\}$, and $x_i = (x_{i1}, x_{i2}, \cdots, x_{id})$ where d is

the dimension of data features, without loss of generality, we assume that $\sqrt{\sum_{j=1}^{d} x_{ij}^2} \leq 1$ where $x_{ij} > 0, y_i \in \{0, 1\}$.

The loss function is then defined as follows:

$$f(\theta) = \frac{1}{n} \sum_{i=1}^{n} [-y_i \log(h_\theta(x_i)) - (1 - y_i) \log(1 - h_\theta(x_i))] \tag{3.1}$$

where h_θ is the output of the sigmoid function given by:

$$h_\theta(x) = \frac{1}{1 + \exp(-\theta^T x)} = \frac{1}{1 + \exp(-\sum_{j=1}^{d} \theta_j x_j)} \tag{3.2}$$

Here we are training the logistic regression model to find the optimal weights $\theta^\star$ that minimizes the loss function $f(\theta)$:

$$\theta^\star = \arg\min_\theta f(\theta) \tag{3.3}$$

When the training process terminates, given a new data point $x = (x_1, x_2, \cdots, x_d)$, the model is able to predict the binary value of the output as follows:

$$y = \begin{cases} 1 & \text{if } h_\theta^\star(x) \geq \tau \\ \\ 0 & \text{if } h_\theta^\star(x) < \tau \end{cases} \tag{3.4}$$

Typically, threshold τ is set as 0.5.

3.2.2 Differential privacy

Definition 1: Given two databases D and D' differing at most one data tuple, let S be a randomized algorithm, and O be the set which contains any possible output of S. S achieves ϵ-*differential privacy*, if and only if the following holds:

$$Pr[S(D) = O] \leq e^{\epsilon} Pr[S(D') = O] \tag{3.5}$$

where ϵ is the privacy budget that is used to control the strength of the privacy guarantee. The privacy preservation of S is strong when the value of ϵ is small.

Laplace mechanism [8] is widely adopted as an efficient way to preserve ϵ-*differential privacy*. In particular, the Laplace mechanism utilizes the global sensitivity Δ, which measures the largest difference in query results within D and D'. ϵ-*differential privacy* is guaranteed by injecting noise η into output of $f(\theta)$ as follows:

$$S(D) = f(D) + \eta, \text{where } \eta \sim Lap(\Delta/\epsilon) \tag{3.6}$$

Since the Laplacian mechanism only works for numerical cases, Mcsherry et al. [12] proposed the Exponential mechanism for achieving ϵ-*differential privacy* in selection process, which is defined as follows:

Definition 2: Let u be a utility function, and Δu be the sensitivity of the utility function. Given a mechanism M and dataset D, we have

$$M(D, u) = \text{choose } r \text{ out of } R \text{ with probability proportional to } \exp(\frac{\epsilon u}{2\Delta u}) \tag{3.7}$$

where R denotes all the possible outcomes.

There are also two properties associated with differential privacy.

Property 1 (Sequential Composition): Let $M_1, M_2, \cdots, M_n$ be a set of mechanism and each satisfies ϵ_i-differential privacy, where $i \in \{1, ..., n\}$. Then the mechanism that sequentially performs $M_1, M_2, \cdots, M_n$ will achieve $\sum_i \epsilon_i-$differential privacy.

Property 2 (Parallel Composition): Assume each M_i achieves ϵ_i-differential privacy, where $i \in \{1, ..., n\}$. A mechanism that performs each M_i over disjoint dataset D_i would

satisfy $\max(\epsilon_i)-$differential privacy.

3.2.3 Layer-wise Relevance Propagation

To measure how related model output is to input features, we adopt Layer-wise Relevance Propagation (LRP) [10, 13] scheme. Specifically, LRP maps the relevances for each layer in a backward sequence for a multi-layer framework. The relevance of nodes in layer l will be calculated from relevance of connected nodes in layer $l+1$. Let R_i^l denote the relevance value of node i in layer l, p_i^l be the activation value of node i in layer l, and w_{ij} be the weight between node i and j, so the relevance backward propagation process could be written as follows:

$$\begin{aligned} a_{ij} &= p_i^l w_{ij} \\ R_i^l &= \sum_j \frac{a_{ij}}{\sum_i a_{ij} + b_l} R_j^{l+1} \end{aligned} \tag{3.8}$$

where b_l is the bias of layer l.

However, when the denominator $\sum_i a_{ij} + b_l$ is too small, the value of R_i^l could be unbounded during the backward propagation process. An efficient way to address this issue is to introduce a predefined stabilizer α as follows:

$$R_i^l = \begin{cases} \sum_j \frac{a_{ij}}{\sum_i a_{ij} + b_l + \alpha} R_j^{l+1} & \text{if } \sum_i a_{ij} + b_l \geq 0 \\ \sum_j \frac{a_{ij}}{\sum_i a_{ij} + b_l - \alpha} R_j^{l+1} & \text{if } \sum_i a_{ij} + b_l < 0 \end{cases} \tag{3.9}$$

where $\alpha \geq 0$.

Hence, the relevance of nodes at each layer could be obtained by applying Eq. 3.8 and 3.9 in the backward propagation process. The relevance between model output $f(\theta)$

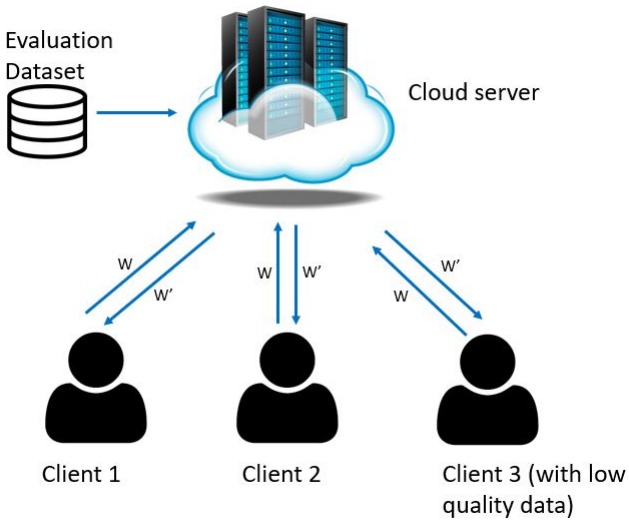


Figure 3.1: Distributed logistic regression with low quality data

and input features can then be described as $R^L_{f(\theta)} = \sum_i R^{l_0}_i$. Finally, we would compute the average relevance $R_j(D)$ for input features x_{ij} and perform linear normalization to range $(0, 1)$ as follows:

$$\begin{aligned} R_j(D) &= \frac{1}{|D|} \sum_{x_i \in D} R^{l_0}_{x_{ij}} \\ R_j(D) &= \frac{R_j(D) - \min_j(R_j(D))}{\max_j(R_j(D)) - \min_j(R_j(D))} \end{aligned} \tag{3.10}$$

3.3 Privacy-Preserving Distributed Logistic Regression

In this section, we introduce the privacy-preserving algorithm for distributed logistic regression with relevance and data quality awareness.

3.3.1 System Architecture

As we can see in Fig. 3.1, there are two types of entity in the system: client and cloud server. Specifically, each client holds a private dataset, and tries to collaboratively train a logistic regression model with other clients. The cloud server coordinates the collaborative

training process at each iteration. Since each client is not willing to share his dataset due to privacy or business concerns, this system allows clients to only share their updated gradients and the cloud server coordinates and averages global parameters accordingly. Besides, there exists an evaluation dataset on the cloud side in case some clients hold low quality data, which might affect the model's performance. The purpose of the evaluation set is to measure the utility score (i.e. accuracy) of the model parameters from each client. Then the cloud server chooses to accept model parameters from a subset of clients with differential privacy guarantee.

We consider the following threat models. Firstly, we assume that the cloud server is honest-but-curious. That is, the cloud server follows the algorithm but tries to infer clients' data records during the training process. Additionally, the clients are also assumed to be honest-but-curious, which means they correctly compute model parameters and upload to the cloud server but try to learn other clients' data records. Clients may also want to learn about the data quality of other clients. Since other clients' data quality information is not a necessity to know, we also aim to protect clients' data quality information from each other. Note that since clients' data quality information is very important for the server to purge low quality data and improve learning accuracy, we do not aim to protect such information from the cloud server. That is, clients' data records are protected against the server but their data quality information is not protected as a type of trade-off between privacy and utility. Multiple clients may collude to learn the private information of other clients.

Since directly sharing parameters with the cloud server would breach data privacy, each client adds Laplacian noise to the polynomial form of the objective function based on the relevance magnitude of different input data features. Then, the noisy gradients are uploaded to the cloud server. Next, on the cloud side, an evaluation set is employed to assess the data quality of clients by testing on their uploaded gradients. A utility score is generated

after assessment. Simply choosing gradients from those clients with high utility scores may also leak data quality privacy. For example, some clients may infer the data quality of other clients if only high data quality clients are selected during the training process. Hence, we adopt the exponential mechanism to protect the privacy of clients' data quality while filtering out low quality data with high probability. In the following, we will discuss the details of proposed privacy-preserving mechanisms.

3.3.2 Differentially Private Relevance

Recall that for multi-layer structure, we can compute the relevance between the input data features and the model output by applying LRP. Based on weights matrix multiplication, sigmoid activation function and unit step function workflow, a logistic regression model can be structured to a multi-layer framework in the following way, and thus LRP can be adopted to compute the relevance between the model output and input features.

In particular, for input layer l_0 of logistic regression model, each dimension j of input x_i has the following relevance rule:

$$R_j^{l_0}(x_i) = \begin{cases} \dfrac{x_{ij}\theta_j}{\sum_{j=1}^{d} x_{ij}\theta_j + b_0 + \alpha} R^{L_s}(x_i) & \text{if } \sum_{j=1}^{d} x_{ij}\theta_j + b_0 \geq 0 \\ \dfrac{x_{ij}\theta_j}{\sum_{j=1}^{d} x_{ij}\theta_j + b_0 - \alpha} R^{L_s}(x_i) & \text{if } \sum_{j=1}^{d} x_{ij}\theta_j + b_0 < 0 \end{cases} \tag{3.11}$$

where α is a predefined stabilizer and $\alpha \geq 0$. b_0 is the bias term. $R^{L_s}(x_i)$ is the relevance propagation rule for sigmoid function layer L_s and defined as follows:

$$R^{L_s}(x_i) = \sum_{j=1}^{d} \frac{1}{1 + \exp(-\theta_j x_{ij})} R^{L}(x_i) \tag{3.12}$$

For output layer L of logistic regression model, relevance of the final output layer L with the unit step function shown in Eq. 3.4 can be rewritten as:

$$R^L(x_i) = \begin{cases} f_{x_i}(\theta) & \text{if } h_\theta^\star \geq \tau \\ \\ 0 & \text{if } h_\theta^\star < \tau \end{cases} \tag{3.13}$$

By substituting Eq. 3.12 to Eq. 3.8 and using Eq. 3.11 at the input layer and Eq. 3.13 at the output layer, the whole relevance between the input feature x_{ij} and the model output would be computed. Hence, we have derived the relevance between the input features and model output for logistic regression.

Since the original input data features are used to compute the relevance value in the LRP scheme, the relevance value might also reveal clients' input data and thus needs to be protected. Hence, we adopt the Laplace mechanism to protect the privacy of original relevance values. In particular, Laplace noise is injected into the $R_j(D)$ to obtain the perturbed relevance $\overline{R}_j$, which is as follows:

$$\overline{R_j} = \frac{1}{|D|} \sum_{x_i \in D} R_{x_{ij}}^{l_0} + Lap(\frac{\Delta}{\epsilon_1}) \tag{3.14}$$

where ϵ_1 is the privacy budget, and Δ is the global sensitivity of relevance. According to Eq. 3.10, $R_{x_{ij}}^{l_0}$ is linearly normalized to range $(0, 1)$, and the number of input features is limited as d, so the largest difference between the relevances of two neighboring dataset is $2d$, and the average relevance difference of dataset D is $2d/|D|$. Therefore, Δ is set to be $2d/|D|$, and the perturbed relevance $\overline{R}_j$ achieves ϵ_1-differential privacy.

3.3.3 Relevance-aware Objective Function Perturbation

In this section, we will perturb the loss objective function of the logistic regression model. Recall that the loss function of logistic regression is $f(x, w) = \log(1 + \exp(x_i^T w)) - y_i x_i^T w$. By applying the Tyler Expansion theorem [14] at the point $x = 0$, we have:

$$\hat{f}(x, w) = \sum_{i=1}^{n} \sum_{k=0}^{\infty} \frac{f_1^{(k)}(0)}{k!} (x_i^T w)^k - \left(\sum_{i=1}^{n} y_i x_i^T \right) w \tag{3.15}$$

According to Eq. 3.15, there exists the infinite summation term and no closed form solution for $f_i^{(k)}(0)$. To address these two issues, the infinite summation is truncated and only reserve the first three orders, i.e. $k = 0, 1, 2$. So, we have $f_1^{(0)} = \log 2, f_1^{(1)} = 1/2, f_1^{(2)} = 1/4$. Then we have the polynomial function $\hat{f}_D(w) = \sum_{i=1}^{n} \sum_{x_i \in D} \gamma_{x_i} \phi(w)$, where γ_{x_i} is the coefficient. Based on functional mechanism [9], the global sensitivity Δ_L can be set as $2 \max_i \sum_{i=1}^{n} |\gamma_{x_i}|$. Thus, we have:

$$\begin{aligned} \Delta_L &= 2 \max \left(\frac{f_1^{(1)}(0)}{1!} \sum_{j=1}^{d} x_j + \frac{f_1^{(2)}(0)}{2!} \sum_{j,l} x_j x_l + y \sum_{j=1}^{d} x_j \right) \\ &\leq 2(d/2 + d^2/8 + d) = d^2/4 + 3d \end{aligned}$$

Next, since some input features are more relevant to the model output and some are less relevant, we introduce more noise to coefficients of the polynomial objective function terms that have less relevant features and less noise to those with more relevant features. According to the privacy-preserving relevance definition in Eq. 3.14, we can divide up the total privacy budget ϵ_2 proportionally to its relevance value, which is shown as follows:

$$\alpha_j = \frac{\overline{R_j}(D)}{\sum_{j=1}^{d} \overline{R_j}(D)}, \forall j \in [1, d]$$
$$\overline{\gamma}_j = \sum_{x_i \in D} \gamma_{jx_i} + Lap(\frac{\Delta_L}{\alpha_j \epsilon_2}) \tag{3.16}$$

Here α_j can be treated as the corresponding contribution of the jth input feature to the model output.

LEMMA 1. Given $\Delta_L = d^2/4 + 3d$, the relevance-aware objective function perturbation achieves ϵ_2-differential privacy.

Proof 1. Let two datasets D and D' be neighboring datasets. Without loss of generality, assume D and D' differ in the last tuple $x_n(x'_n)$. Since the perturbation of the relevance $R_j(D)$ can be written as Eq. 3.14, we have:

$$\begin{aligned}
\frac{Pr(\overline{f}(D))}{Pr(\overline{f}(D'))} &= \frac{\prod_{i=1}^{n}\prod_{j=1}^{d} \exp(\frac{\alpha_j \epsilon_2 \| \sum_{x_i \in D} \gamma_{jx_i} - \overline{\gamma}_j \|}{\Delta_L})}{\prod_{i=1}^{n}\prod_{j=1}^{d} \exp(\frac{\alpha_j \epsilon_2 \| \sum_{x'_i \in D'} \gamma_{jx'_i} - \overline{\gamma}_j \|}{\Delta_L})} \\
&\leq \prod_{i=1}^{n}\prod_{j=1}^{d} \exp(\frac{\alpha_j \epsilon_2 \parallel \sum_{x_i \in D} \gamma_{jx_i} - \sum_{x'_i \in D'} \gamma_{jx'_i} \parallel}{\Delta_L}) \\
&\leq \prod_{i=1}^{n}\prod_{j=1}^{d} \exp(\frac{\alpha_j \epsilon_2 \parallel \gamma_{jx_n} - \gamma_{jx'_n} \parallel}{\Delta_L}) \\
&\leq \exp(\frac{\epsilon_2 \sum_{i=1}^{n}\sum_{j=1}^{d} \frac{\overline{R_j}(D)}{\sum_{j=1}^{d} \overline{R_j}(D)} \parallel \gamma_{jx_n} - \gamma_{jx'_n} \parallel}{\Delta_L}) \\
&\leq \exp(\frac{2\epsilon_2 \max_i \sum_{i=1}^{n} \parallel \gamma_{x_i} \parallel}{\Delta_L}) \\
&= \exp(\epsilon_2)
\end{aligned} \tag{3.17}$$

This concludes the proof. □

3.3.4 Privacy-Preserving Selection

In the system, some clients may hold low-quality data, which is quite possible in reality. Take medical data records for an example. Small hospitals may not have advanced equipment to gather as accurate data as large hospitals. To reduce the possible impact of low-quality data on learning accuracy, we propose a data quality-aware selection mechanism on the cloud side to purge the effect of low-quality data. To ensure that clients cannot infer other clients' data quality information, the selection processes provides differential privacy guarantee.

In particular, an evaluation dataset is utilized by the cloud server to assess each client's data quality. Each client's uploaded gradients are evaluated with the dataset and a utility score is generated. For simplicity, we use the publicly available benchmark dataset MNIST [15] as the evaluation dataset (in deployment, other datasets could be used dependent on the application scenarios) and classification accuracy is used as the utility score. If the server directly chooses the clients whose model gradients have high utility scores, that could potentially leak some clients' data quality information to other clients via the aggregate model parameters. To address this, we adopt the exponential mechanism to add uncertainty to this selection process. Specifically, the server samples M clients without replacement such that

$$Pr(\text{choose client } i) \propto \exp(\frac{\epsilon_3 u_i}{2M\Delta u}) \tag{3.18}$$

where u_i is the utility score of client i, and Δu is the global sensitivity.

LEMMA 2. Given a dataset D, the global sensitivity Δu for classification tasks would be $1/2$.

Proof 2. Let x and y be number of correct predictions and total number of records, respec-

tively. Since $y \geq 1$ and $y \geq x$, so we have:

$$\begin{aligned} \Delta u &= \frac{x+1}{y+1} - \frac{x}{y} \\ &= \frac{y-x}{y(y+1)} \leq 1/2 \end{aligned} \tag{3.19}$$

Thus, we set $\Delta u = 1/2$, and this concludes the proof.

□

LEMMA 3. The selection process on the cloud side satisfies ϵ_3-differential privacy.

Proof 3. According to Definition 2, selecting each client i satisfies $\frac{\epsilon_3}{M}-$differential privacy. The whole selection process samples M clients which would achieve ϵ_3-differential privacy.

□

The whole privacy-preserving logistic regression scheme is summarized in Algorithm 3. On the client side, differentially private relevance and objective function perturbation achieve $(\epsilon_1 + \epsilon_2)-$differential privacy for data records. On the cloud side, privacy-preserving selection satisfies ϵ_3-differential privacy for data quality information.

3.4 Performance Evaluation

In this section, we evaluate the performance of our differentially private logistic regression algorithm with relevance and data quality awareness, which is abbreviated as DPLRRQ. We use two well-known benchmark datasets: Integrated Public Use Microdata Series (IPUMS)[16] and Default of Credit Card Clients [17]. There are 600,000 census records in the IPUMS dataset, the attributes of which include *Sex, Age, Race, Family Size, Number of Children, Ownership of Dwelling, Living Difficulty, Education, Hours Work per Week, Filed of Degree, Number of Children, Number of Rooms, Private Health Insurance, Marital*

Algorithm 3 Privacy-preserving distributed logistic regression algorithm

Input: Database $D_1, D_2, \cdots, D_N$, loss function $L(\theta)$, privacy budget $\epsilon_1, \epsilon_2, \epsilon_3$, sampling size M, the number of batches T, the number of clients N, random seed S

Output: θ^T

The cloud server initializes all the global parameters with random seed S
Each client downloads the current global parameters
Each client computes $R_j(D) = \frac{1}{|D|}\sum_{x_i \in D} R_{x_{ij}}(x_i)$ for j in $[1, d]$ according to the LRP Algorithm (3.10)
Set $\Delta = 2d/|D|$ based on Section 3.3.2
for $j \in [1, d]$ **do**
    $\overline{R_j} = \frac{1}{|D|}\sum_{x_i \in D} R_{x_{ij}}(x_i) + Lap(\frac{\Delta}{\epsilon_1})$
end for
$\alpha_j = \frac{\overline{R_j}(D)}{\sum_{j=1}^{d} \overline{R_j}(D)}, \forall j \in [1, d]$
Convert loss function $f(x, w)$ to approximated loss function $\hat{f}_i(x, w)$
$\hat{f}_i(x, w) = \sum_{i=1}^{n} \sum_{k=0}^{2} \frac{f_1^{(k)}(0)}{k!}(x_i^T w)^k - \left(\sum_{i=1}^{n} y_i x_i^T\right) w$
Set $\Delta_L = d^2/4 + 3d$ based on Lemma 1
for $j = 1$ to d **do**
    $\overline{\gamma}_j = \sum_{x_i \in D} \gamma_{j x_i} + Lap(\frac{\Delta_L}{\alpha_j \epsilon_2})$
end for
for $t \in [T]$ **do**
    for $i = 1$ to N **do**
        Each client i computes $\theta_i^t = \arg\min \overline{f}_i(w)$
        The cloud server collects all the model parameters θ_i^t and choose to accept M clients by applying (3.18)
        The cloud server averages M model parameters, updates and broadcasts the global parameters
    end for
end for
return θ^T

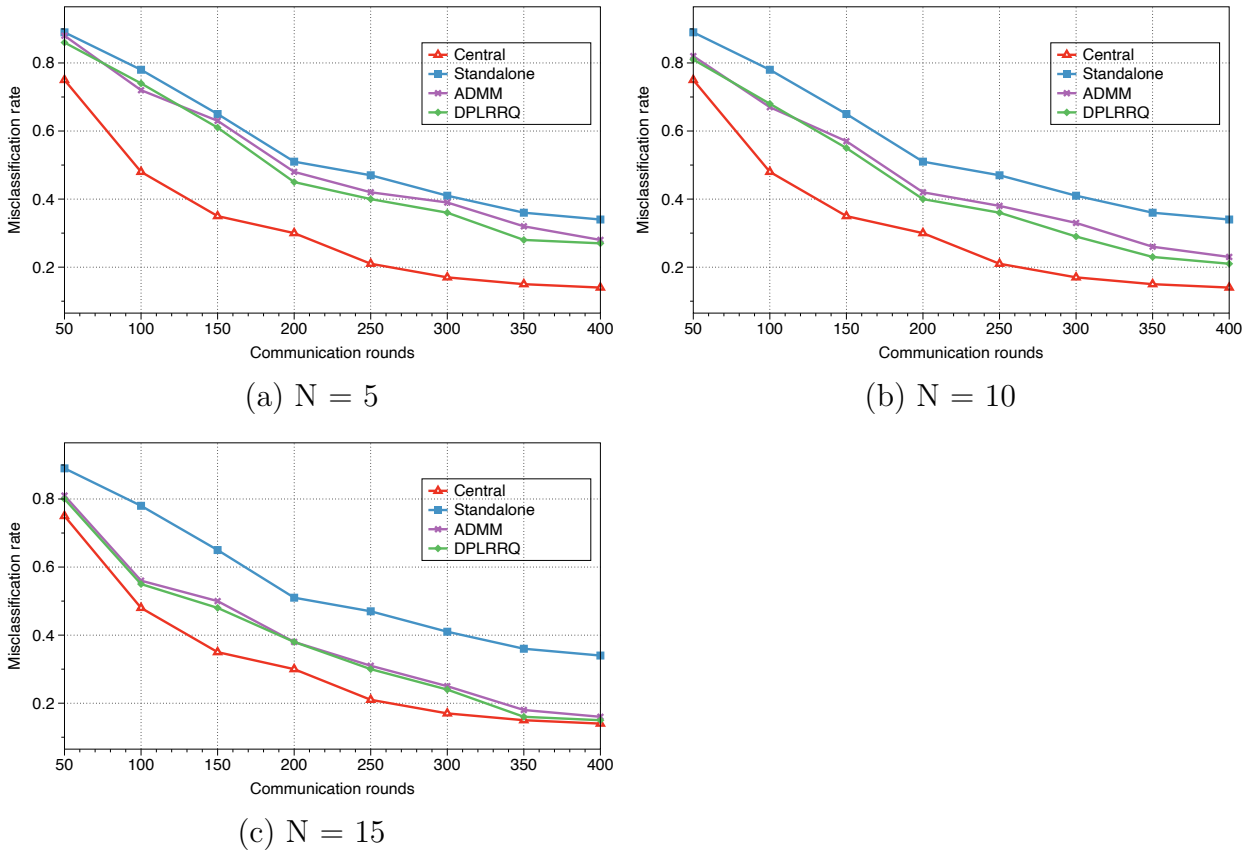
Status, and *Income Class*. The logistic regression task based on this dataset is to classify whether the person belongs to the high-income class or not. For the Default of Credit Card Clients dataset, it contains 30,000 records and 24 attributes, including *credit card owner attributes (e.g., gender), history of past payment, amount of bill statement attributes, and amount of previous payment attributes*. The classification goal of Credit Card dataset is to predict whether a credit user will default or not.

For categorical features in the two datasets, we use the one-hot encoding [18] to transform these attributes to numerical values. All the other feature values are normalized to range [0, 1]. We randomly sample 20% of the records as the testing set, 10% of the records as the evaluation set, and the remaining records as the training dataset. Then training dataset is randomly split into 5 partitions and each client holds one partition. N is the total number of clients. The clients hold the partitions evenly, such that each partition is held by $N/5$ clients. We use the misclassification rate as the performance metric, which is defined as the ratio of falsely classified records to total records.

To simulate the low-quality data held by some clients, we randomly choose 1/3 of clients and replace a random fraction (denoted by β) of their dataset with random noise drawn from range [0, 1]. For simplicity, in the following, we set $\epsilon_1 = \epsilon_2 = \frac{1}{2}\epsilon_3$, and use ϵ to denote ϵ_3.

3.4.1 Compared Models

We compare our proposed scheme with other benchmark schemes. The first one is the *Central* framework, where all the data are collected and trained in a centralized manner without protecting privacy. The second one is *ADMM* [19], where each client adds differentially private noise to the uploaded parameters in each iteration. The last one is the *standalone* framework. No distributed learning is involved in the standalone framework, and



(a) N = 5

(b) N = 10

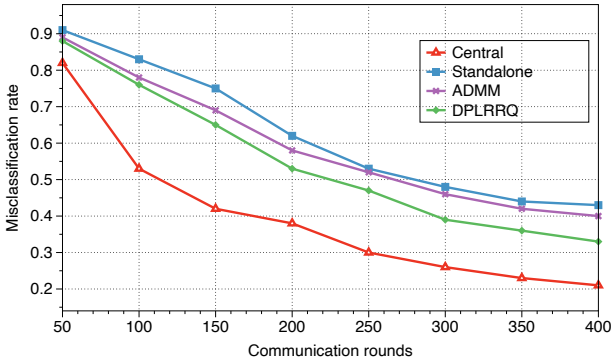
(c) N = 15

Figure 3.2: Training convergence for the IPUMS dataset with different number of clients

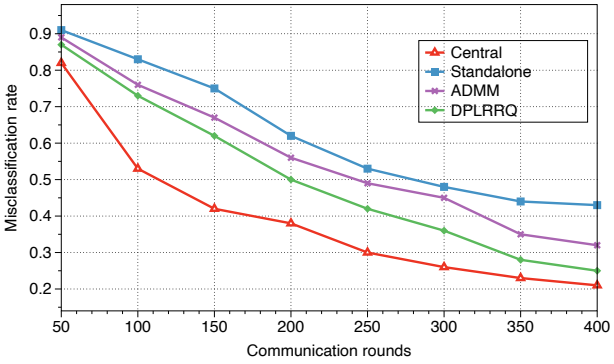
each client learns based on its own data without considering privacy. All the frameworks are implemented on Pytorch [20].

3.4.2 Training Convergence

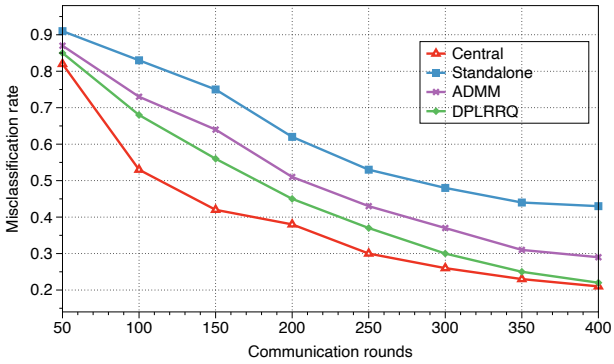
In this experiment, we measure the training convergence with different frameworks. The number of clients ranges from 5 to 15. We set $\beta = 0, \epsilon = 1$ and $M = 0.6N$. As illustrated in Fig. 3.2 and Fig. 3.3, our proposed DPLRRQ outperforms ADMM and Standalone, which is especially clear over the Credit Card dataset. DPLRRQ outperforms ADMM because it considers the relevance between input features and the model output which reduces the noise added to the model and it also considers the data quality of clients which mitigates the impact of low-quality data. Intuitively, the Standalone scheme achieves the worst misclassification



(a) N = 5



(b) N = 10



(c) N = 15

Figure 3.3: Training convergence for the Credit Card dataset with different number of clients

rate due to the limited training data that each client has. The Central scheme has the best performance since it has all the data and it does not add noise for differential privacy. When the number of clients increases, however, the performance of our DPLRRQ scheme gets closer to that of the the Central scheme. This is mainly because more data is available for training at each iteration when there are more clients.

3.4.3 Data Quality

In this part, we evaluate the data quality awareness of the proposed scheme. To better demonstrate the effectiveness of using an evaluation dataset at the cloud server in our scheme, we add *No Evaluation* into comparison, which is the same as DPLRRQ except that it does not have the privacy-preserving selection based on the evaluation dataset (see Section 3.3.4). Fig. 3.4 and Fig. 3.5 show the misclassification rates of different schemes when the fraction of low-quality data $\beta = 20\%, 40\%, 60\%$. Generally, the misclassification rate increases as the amount of low-quality data increases. It is worth noting that there is minimal degradation of performance in our scheme, while all the other frameworks are significantly impacted. For example, at a high low-quality data level $\beta = 60\%$ in the IPUMS dataset, the misclassification rate for DPLRRQ is 23%, while for Central, Standalone, ADMM and No Evaluation, the misclassification rate reaches 38%, 45%, 44% and 44% respectively. This shows the effectiveness of our scheme in considering data quality.

3.4.4 Accuracy vs. Privacy Budget

In this experiment, the classification accuracy of different schemes under various privacy budgets are evaluated. To show the effectiveness of considering the relevance between input features and the output model in our proposed scheme DPLRRQ, we added *No Relevance* into comparison, which is the same as DPLRRQ except that the relevance is not

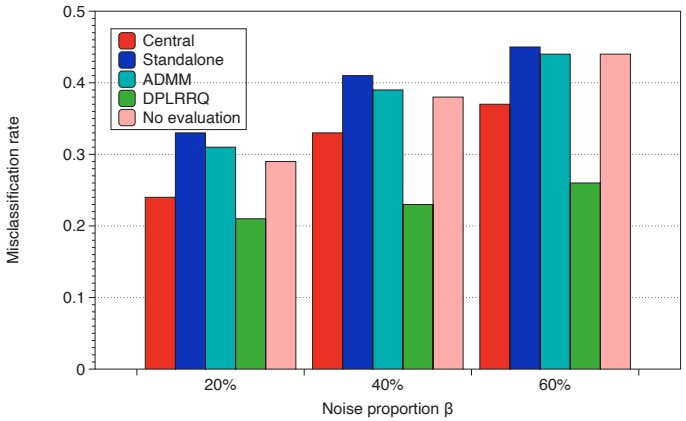


Figure 3.4: Misclassification rate of different frameworks with varying noise proportion (IPUMS, N = 15)

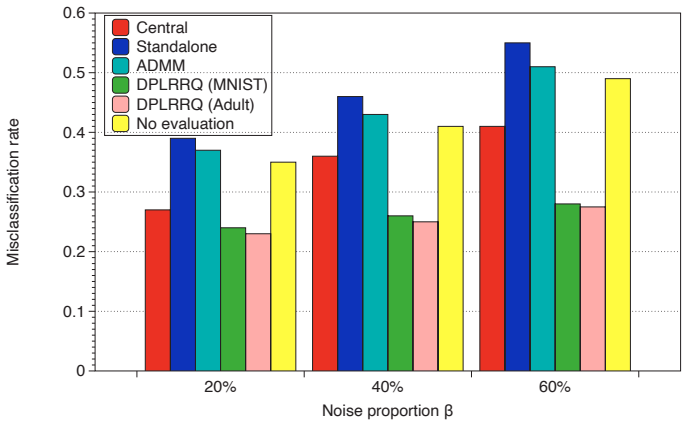


Figure 3.5: Misclassification rate of different frameworks with varying noise proportion (Credit Card, N = 15)

considered and the same scale of noise is added to all the coefficients of the objective function. As shown in Fig. 3.6 and Fig. 3.7, the misclassification rate of DPLRRQ, ADMM and No Relevance decreases as the privacy budget increases. DPLRRQ outperforms ADMM and No Relevance due to the consideration of relevance. Central and Standalone are not affected by privacy budgets since they do not provide differential privacy. Our DPLRRQ scheme reaches comparable misclassification rate with Central when $\epsilon = 1$ or higher, which means our scheme can achieve high accuracy while maintaining privacy. In addition, our scheme outperforms Standalone when $\epsilon \geq 0.1$.

To summarize, our proposed DPLRRQ achieves a high model accuracy and a strong

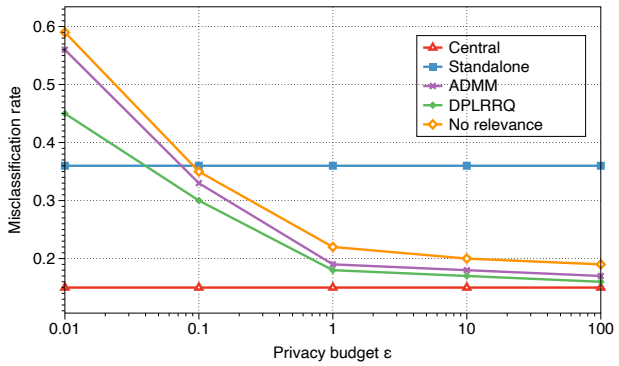


Figure 3.6: Misclassification rate under different privacy budgets (IPUMS, N = 15)

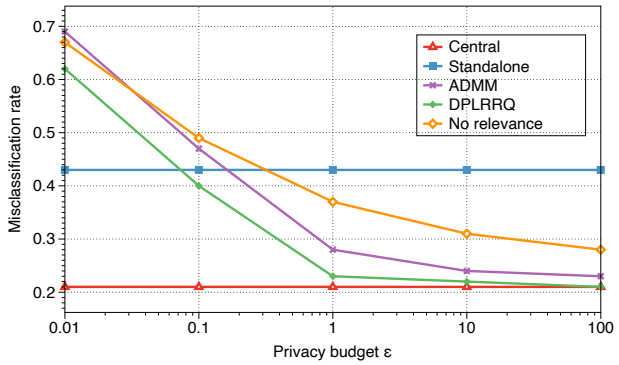


Figure 3.7: Misclassification rate under different privacy budgets (Credit Card, N = 15)

privacy guarantee.

3.5 Related Work

There is a line of work on privacy-preserving logistic regression. Zhang et al. [21] protected the training data via homomorphic encryption. Bost et al. [22] investigated several classifiers in the setting of two-party computation, in which the server holds the secret model and the client holds the sensitive data. Both interact in a way with many rounds so that at the end the client learns the classifier with high accuracy. Recently, Jayaraman et al. [23] proposed to improve the noise bounds of logistic learning by adding noise directly to the aggregated model parameters in the multiparty computation setting. Duverle et al. [24] also proposed a secure two-party learning algorithm to train the logistic regression classifier without revealing private training sets from each client. Xie et al. [25] showed a secure

framework to train a logistic regression model in a distributed manner. Their method is based on Yao's garbled circuit and an additive homomorphic encryption scheme. Kim et al. [26, 27] demonstrated secure outsourcing methods to train a logistic regression model on encrypted data and showed their feasibility with real datasets. However, these encryption-based schemes generally incur high computation and communication costs.

Zhang et al. [9] demonstrated the Functional mechanism (FM) that protects user data by perturbing the coefficients of the objective function and finds the optimal parameters by minimizing the perturbed objective function. However, the noise injected to coefficients are the same, which may have negative impact on model accuracy. Ligett et al. [28] proposed a general noise reduction framework for regularized linear regression based on the covariance perturbation and output perturbation. In particular, the noise reduction mechanism is adopted to generate a variety of private hypothesis by gradually relaxing the value of ϵ and the privacy hypothesis can be computed by optimizing the perturbed objective function. Recently, Du et al. [29] designed a privacy-preserving logistic regression training framework by adding carefully-crafted noise to objective function. In addition, other works adopting different types of noise scaling to achieve differential privacy over distributed data have also been reported [30, 31, 32, 33]. However, the existing work has not addressed privacy-preserving distributed logistic regression considering the relevance between input features and the output model and the data quality of clients.

3.6 Summary

In this chapter, we investigated privacy-preserving distributed logistic regression. We developed differentially private layer-wise relevance propagation and loss function perturbation for logistic regression models, which can achieve high accuracy due to consideration of the relevance between input data features and the output model. When generating the

aggregate model, the cloud server takes clients' data quality into account through an evaluation dataset while protecting clients' data quality privacy via the exponential mechanism. Experimental results showed that the proposed differentially private learning scheme can achieve low misclassification rate and strong robustness against low-quality data.

Bibliography

[1] S. B. Kotsiantis, I. Zaharakis, and P. Pintelas, "Supervised machine learning: A review of classification techniques," *Emerging artificial intelligence applications in computer engineering*, vol. 160, pp. 3–24, 2007.

[2] F. N. Motlagh and H. Sajedi, "Mosar: a multi-objective strategy for hiding sensitive association rules using genetic algorithm," *Applied Artificial Intelligence*, vol. 30, no. 9, pp. 823–843, 2016.

[3] Y. Li, Z. L. Jiang, L. Yao, X. Wang, S. Yiu, and Z. Huang, "Outsourced privacy-preserving c4. 5 decision tree algorithm over horizontally and vertically partitioned dataset among multiple parties," *Cluster Computing*, vol. 22, no. 1, pp. 1581–1593, 2019.

[4] G. Dudek, "Pattern-based local linear regression models for short-term load forecasting," *Electric Power Systems Research*, vol. 130, pp. 139–147, 2016.

[5] T. Li, J. Li, Z. Liu, P. Li, and C. Jia, "Differentially private naive bayes learning over multiple data sources," *Information Sciences*, vol. 444, pp. 89–104, 2018.

[6] X. Liu, X. Zhu, M. Li, L. Wang, E. Zhu, T. Liu, M. Kloft, D. Shen, J. Yin, and W. Gao, "Multiple kernel k-means with incomplete kernels," *IEEE transactions on pattern analysis and machine intelligence*, 2019.

[7] C. Dwork, "Differential privacy," *Encyclopedia of Cryptography and Security*, pp. 338–340, 2011.

[8] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Theory of cryptography conference*, pp. 265–284, Springer, 2006.

[9] J. Zhang, Z. Zhang, X. Xiao, Y. Yang, and M. Winslett, "Functional mechanism: regression analysis under differential privacy," *arXiv preprint arXiv:1208.0219*, 2012.

[10] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek, "On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation," *PloS one*, vol. 10, no. 7, p. e0130140, 2015.

[11] L. Zhao, Q. Wang, Q. Zou, Y. Zhang, and Y. Chen, "Privacy-preserving collaborative deep learning with unreliable participants," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 1486–1500, 2019.

[12] F. McSherry and K. Talwar, "Mechanism design via differential privacy.," in *FOCS*, vol. 7, pp. 94–103, 2007.

[13] N. Phan, X. Wu, H. Hu, and D. Dou, "Adaptive laplace mechanism: Differential privacy preservation in deep learning," in *2017 IEEE International Conference on Data Mining (ICDM)*, pp. 385–394, IEEE, 2017.

[14] G. B. Arfken and H. J. Weber, "Mathematical methods for physicists," 1999.

[15] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[16] M. Center, "Integrated public use microdata series, international: Version 6.4 [database]," *University of Minnesota, Minneapolis http://doi. org/10.18128/D020V64*, 2015.

[17] I.-C. Yeh and C.-h. Lien, "The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients," *Expert Systems with Applications*, vol. 36, no. 2, pp. 2473–2480, 2009.

[18] M. Cassel and F. Lima, "Evaluating one-hot encoding finite state machines for seu reliability in sram-based fpgas," in *12th IEEE International On-Line Testing Symposium (IOLTS'06)*, pp. 6–pp, IEEE, 2006.

[19] Y. Hu, P. Liu, L. Kong, and D. Niu, "Learning privately over distributed features: An admm sharing approach," *arXiv preprint arXiv:1907.07735*, 2019.

[20] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.

[21] J. Zhang, R. Jin, Y. Yang, and A. G. Hauptmann, "Modified logistic regression: An approximation to svm and its applications in large-scale text categorization," in *ICML*, vol. 3, pp. 888–895, 2003.

[22] R. Bost, R. A. Popa, S. Tu, and S. Goldwasser, "Machine learning classification over encrypted data.," in *NDSS*, vol. 4324, p. 4325, 2015.

[23] B. Jayaraman and L. Wang, "Distributed learning without distress: Privacy-preserving empirical risk minimization," *Advances in Neural Information Processing Systems*, 2018.

[24] D. A. Duverle, S. Kawasaki, Y. Yamada, J. Sakuma, and K. Tsuda, "Privacy-preserving statistical analysis by exact logistic regression," in *2015 IEEE Security and Privacy Workshops*, pp. 7–16, IEEE, 2015.

[25] W. Xie, Y. Wang, S. M. Boker, and D. E. Brown, "Privlogit: Efficient privacy-preserving logistic regression by tailoring numerical optimizers," *arXiv preprint arXiv:1611.01170*, 2016.

[26] M. Kim, Y. Song, S. Wang, Y. Xia, and X. Jiang, "Secure logistic regression based on homomorphic encryption: Design and evaluation," *JMIR medical informatics*, vol. 6, no. 2, p. e19, 2018.

[27] A. Kim, Y. Song, M. Kim, K. Lee, and J. H. Cheon, "Logistic regression model training based on the approximate homomorphic encryption," *BMC medical genomics*, vol. 11, no. 4, pp. 23–31, 2018.

[28] K. Ligett, S. Neel, A. Roth, B. Waggoner, and S. Z. Wu, "Accuracy first: Selecting a differential privacy level for accuracy constrained erm," in *Advances in Neural Information Processing Systems*, pp. 2566–2576, 2017.

[29] W. Du, A. Li, and Q. Li, "Privacy-preserving multiparty learning for logistic regression," in *International Conference on Security and Privacy in Communication Systems*, pp. 549–568, Springer, 2018.

[30] Y. Fan, J. Bai, X. Lei, Y. Zhang, B. Zhang, K.-C. Li, and G. Tan, "Privacy preserving based logistic regression on big data," *Journal of Network and Computer Applications*, vol. 171, p. 102769, 2020.

[31] L. Song, H. Wu, W. Ruan, and W. Han, "Sok: Training machine learning models over multiple sources with privacy preservation," *arXiv preprint arXiv:2012.03386*, 2020.

[32] S. Suthaharan, "Characterization of differentially private logistic regression," in *Proceedings of the ACMSE 2018 Conference*, pp. 1–8, 2018.

[33] J. M. Cortés-Mendoza, A. Tchernykh, M. Babenko, L. B. Pulido-Gaytán, G. Radchenko, F. Leprevost, X. Wang, and A. Avetisyan, "Privacy-preserving logistic regression as a cloud service based on residue number system," in *Russian Supercomputing Days*, pp. 598–610, Springer, 2020.

4 Privacy-Preserving Cloud-Assisted Efficient Federated Learning

4.1 Introduction

In recent years, more and more data are generated by individuals in various domains. Industries and academia have been developing tools and techniques to transform data into useful insights aimed to improve various aspects of our life [1]. Machine learning techniques have gained increasing popularity over years to extract knowledge from data. For example, massive data from social media websites and apps have been collected every year, such as browse histories in Twitter, which facilitates analysis and sending customized contents to targeted customers. Machine learning algorithms also have massive applications in the medical field. For example, hospitals have been adopting advanced computer vision devices to detect cancer based on the images. This could provide high detection accuracy and easy to scale.

Traditionally, a typical machine learning algorithm is to learn and make predictions based on a single dataset [2]. However, a number of data resources are increasingly distributed and owned by different organizations. For example, financial data can be distributed in several banks and financial institutes. Pooling data from multiple sources for learning can usually achieve better prediction performance. Consequently, the traditional paradigm of learning from a single dataset has been shifting towards distributed learning, i.e., data from multiple parties are used to collaboratively train a learning model. In a conventional collaborative learning approach, there exists a central server, i.e., a cloud server and let multiple data owners directly upload their data to the cloud server for training [3, 4].

Although federated learning achieves better performance than single dataset based learning, there are some concerns on the data privacy. It is likely that during the training

process, the private information of each party, e.g. health data records, can be disclosed, which will breach users' privacy. Hitaj et al. [5] proposed a GAN-based reconstruction attack against the federated learning by assuming a malicious client, which utilized the shared model as the discriminator to train a GAN framework. Hence, it becomes increasingly critical to design a protocol to train a learning model from the distributed datasets, while guarantee their privacy.

In addition, most federated learning frameworks require massive training data from clients. Clients are usually distributed and may perform their unique activities associated with their own environment, which leads to local dataset generated with different sizes and distributions. In other words, clients generally holds non-IID dataset. Moreover, mobile clients usually have limited resources and could not participate in time-consuming and communication-extensive training process. Thus, efficiency aspect of federated learning also needs to be addressed.

In this chapter, we propose an efficient privacy-preserving federated learning algorithm. Specifically, each client trains on his local dataset and select the gradients that are *aligned* with global model gradient tendency, as some local updates may not contribute to the model convergence due to non-IID data among clients. Gradient magnitude is also considered as another selection criteria as large magnitude usually means more impact to model training. By excluding these less consistent gradient uploads, the communication overhead could be reduced significantly. In addition, clients will first add noise to perturb the gradients uploading to cloud server. Then, homomorphic encryption is adopted to encrypt the noisy gradients before uploading to cloud server. Finally, the cloud server decrypts the sum of the noisy gradients and updates the global model parameters without learning anything else.

The main contributions of this chapter are summarized as follows:

- We propose an efficient privacy-preserving federated learning framework, which carefully selects aligned and large magnitude gradients for uploading and in this way significantly accelerates convergence and reduces the communication overhead during training.

- We propose a privacy-preserving gradient uploading scheme that combines distributed and collective noise adding and efficient homomorphic encryption to achieve differential privacy without incurring too much noise. In particular, each client's data is protected by adding a small amount of noise to selected gradients before uploading. Noisy gradients are then encrypted so that the cloud server will only get the noisy sum of gradients. The noisy sum contains minimal but enough noise to achieve $\epsilon-$differential privacy. Our scheme does not rely on the cloud server or a third party to add noise; the clients collaboratively generate noise for differential privacy.

- Extensive experimental results show that the proposed scheme can achieve high accuracy while dramatically reduces communication overhead with strong privacy guarantee.

The rest of the chapter is organized as follows. Section 4.2 presents preliminaries for distributed SGD and differential privacy. Section 4.3 introduces proposed framework. Performance evaluation is presented in Section 4.4. Section 4.5 reviews related work. Section 4.6 concludes this chapter.

4.2 Preliminaries

4.2.1 Distributed SGD

Deep learning has been widely adopted to extract useful knowledge in image recognition, text mining, and audio inferences areas. Stochastic gradient descent (SGD) [6] is

proposed and proven to be an efficient way to train deep learning frameworks. In particular, with a mini-batch of client data, the model first computes the objective loss function $L(\theta)$ based on the difference of predicted label y' and ground truth label y. Then, the loss function would be minimized by updating the weights of model through $-\partial L/\partial\theta$ direction. The training process iterates until convergence.

Distributed SGD [7] has been proposed to minimize the objective loss function $L(\theta)$ in distributed ways. Applying classic mini-batch SGD, we perform updates to the model parameters θ in the following way. Let $\alpha_n \in D_n$ be a mini-batch dataset of client n, then at each iteration k, distributed SGD can be performed as follows:

$$\theta^{k+1} := \theta^k - \eta[\frac{1}{N}\sum_{n\in N} g(\theta^k)] \tag{4.1}$$

where each client n computes his own gradient $g(\theta) = (1/|\alpha_n|)\sum_{i\in\alpha_n}\Delta l(\theta; x_i^n, y_i^n)$ based on the mini-batch α_n, and the cloud server averages these gradients from all the clients and updates the global model parameters with learning rate η.

4.3 System Framework

In this section, we will introduce the system architecture, efficient gradient upload and differentially-private uploading scheme under the federated learning framework.

4.3.1 System Overview

The federated learning framework for deep learning with edge computing is depicted in Fig. 4.1. The system consists of two different entities: client node and cloud server. Client nodes include different kinds of devices, such as smartphones, laptop, and wearable devices.

Specifically, each client node holds his own private dataset D, and is willing to col-

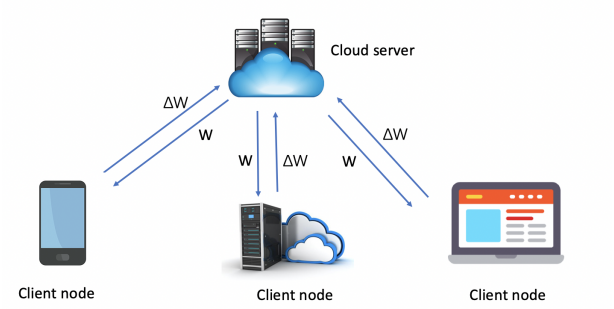


Figure 4.1: Federated learning framework

laboratively train a deep learning model without revealing his own data. Generally mobile clients are resource-limited. Thus, the local gradients are selected based on the efficient gradient upload scheme. Moreover, local gradients are perturbed with differentially-private noise and encrypted before uploading to prevent the cloud server from learning additional information (e.g. each client's data). The noisy sum aggregated by the cloud server will not change much whether a client participates or not.

Despite the federated learning system benefits significantly from edge computing, there are mainly two challenges existing in the system: a) during training process, how to reduce communication overhead while ensure high model utility; b) how to securely aggregate each client's gradients with leaking his data privacy. To address those challenges, we propose an efficient gradient selection scheme and privacy-preserving parameter updating mechanism respectively.

4.3.2 Threat Model

We assume an honest-but-curious cloud server. Namely, the cloud server will correctly execute operations according to the algorithm, but is curious to learn any information from the clients. Additionally, some clients may be compromised during training to collude and steal sensitive information from honest clients. It is assumed that the system has an estimate over the upper bound of β for the fraction of compromised clients.

A trusted key dealer is also assumed to securely issue keys to the clients and the cloud server. Note that this assumption can be easily relaxed to an honest-but-curious key dealer not colluding with the cloud server.

4.3.3 Local Gradient Selection

Motivation

In federated learning domain, the learning model is obtained by pooling massive distributed local models. Since local models are trained using the client-specific data, to some extent, there exists difference between the local and global models. As we mentioned the non-IID property of local data, some local updates may not contribute to the training process. For example, when training a computer vision framework on mobile, clients usually have different local datasets associated with his own environment and usage pattern. Some local updates could be tangential to the collaborative trend of the training convergence. Thus, uploading these local gradients to the cloud server makes little contributions and may also slow down the convergence of the training process.

Communication-efficient Upload

Since some local updates may not contribute to the convergence of training process, we need to find a way to identify and filter out these local updates. The intuition here is to compare the local updates with global trend. Specifically, the global gradient direction indicates the current global convergence trend, if the local updates could not align well with the global direction, then there is no need to upload the current local updates to the cloud server.

Gradient magnitude has been used to estimate the relevance of the local updates in many work ([9], [10]). However, even though magnitude could be an good estimate of local

updates, it may still not identify and filter out those helpless local gradients. Specifically, some local updates may contain gradients with large norm magnitude, but their gradient direction may not be consistent with global model trend, therefore contributes little to global convergence. To the contrary, some local updates with small norm magnitude could have significant impact to global convergence if their gradient direction aligns well with global trend.

Algorithm 4 Communication-efficient federated learning

Input: Database $D_1, D_2, \cdots, D_N$, mini-batch size B, alignment threshold ω_λ, magnitude threshold ω_m, learning rate η, number of gradients M, bound C, privacy budget ϵ_1

Output: θ^T

Initialize global parameters and global gradient $\overline{\mathbf{g}}^0$
for each iteration $t = 1, \cdots, T$ **do**
  for all client $i = 1, 2, \cdots, N$ in parallel **do**
    Client i downloads current global model parameters θ^t
    Client i computes the gradients $\mathbf{g}_i^t$ based on his own dataset
    if $\lambda(\mathbf{g}_i^t, \overline{\mathbf{g}}^{t-1}) + Lap(\frac{4M}{\epsilon_1}) \geq \omega_\lambda^t + Lap(\frac{2M}{\epsilon_1})$ or $|\mathbf{g}_i^t| + Lap(\frac{8MC}{\epsilon_1}) \geq \omega_m^t + Lap(\frac{4MC}{\epsilon_1})$ according to Eq. 4.3 and 4.4 **then**
      Client i uploads $\mathbf{g}_i^t$ to cloud server
      Cloud server adds $\mathbf{g}_i^t$ to set κ
    end if
  end for
  Cloud server computes the average of uploaded gradients: $\overline{\mathbf{g}}^t = \frac{1}{|\kappa|} \sum_{\mathbf{g}_i^t \in \kappa} \mathbf{g}_i^t$
  $\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^t - \eta \overline{\mathbf{g}}^t$
end for
return $\boldsymbol{\theta}^T$

To this end, we propose an efficient communication upload scheme which considers not only the magnitude, but also the local gradient direction alignment with the current global convergence trend. Specifically, in each learning iteration, clients compare their local updates with the global update so as to determine if their updates are aligned with global update. The challenge here is that the global update cannot be known before the cloud aggregation happens. One way to address this is to use the global gradients from previous iteration to estimate the current global gradients, because there usually exists only slightly

change between two consecutive updates.

Next, we propose an efficient way to measure the alignment of local update and global update. In particular, we compare the sign of the local gradients and global gradients element-wise, as same sign of gradients indicates similar tendency of updates. For each iteration t, the metric is described as follows:

$$\psi(g_{ij}^t, \overline{g}_j^t) = \begin{cases} 1, & sign(g_{ij}) = sign(\overline{g}_{ij}) \\ \\ 0, & \text{otherwise} \end{cases} \tag{4.2}$$

$$\lambda(\mathbf{g}_i^t, \overline{\mathbf{g}}^t) = \frac{1}{M} \sum_{j=1}^{M} \psi(g_{ij}^t, \overline{g}_j^t) \tag{4.3}$$

where $\mathbf{g}_i^t$ and $\overline{\mathbf{g}}^t$ mean the local gradient for client i and global gradient at iteration t, respectively. M is the length of the gradients.

If the value of alignment metric in Eq. 4.3 for local update and global update is less than a pre-defined threshold ω_λ^t, where t is the iteration index, the local update will not be uploaded to the cloud server since it's not consistent with the tendency of global convergence. In this way, we can significantly reduce the communication overhead in the training process.

In addition, for gradient magnitude, we also select the local gradients which are greater than a pre-defined threshold to upload, as large magnitude would have significant contribution to global convergence. The whole selection process of each client i at iteration t is shown as follows:

$$Selection_i^t = \{\mathbf{g}_i^t |\ \lambda(\mathbf{g}_i^t, \overline{\mathbf{g}}^t) \geq \omega_\lambda^t \text{ or } |\mathbf{g}_i^t| \geq \omega_m^t\} \tag{4.4}$$

Since how gradients are selected for sharing may also leak clients' privacy, we adopt sparse vector technique [11] to protect the threshold selecting process. The details of the

procedure is summarized in Algorithm 4.

4.3.4 Privacy-preserving Parameter Update

Since the model training process requires gradients sharing between clients and cloud server, the cloud server might learn each client's gradients during training process which will breach clients' privacy. To this end, we propose a privacy-preserving parameter update scheme. In particular, clients will first perturb the gradients uploading to cloud server by adding Laplacian noise. Then, the noisy gradients will be encrypted with homomorphic encryption before uploading to cloud server. Finally, the cloud server decrypts the sum of the noisy gradients and updates the global model parameters.

There are three privacy aspects being considered in our scheme. First, the cloud server gets the noisy sum only but nothing else (e.g. each individual client's data). Second, an client will not infer anything without the cloud sever capability. The last privacy aspect is that the noisy sum obtained by cloud server will not be affected much whether a specific client participate in the training or not. We will describe the details of encryption method and achieving differential privacy in the following sections.

Encryption method [12]

Setup: The key dealer first creates a set of Nq random secrets, i.e. $T = \{t_1, \cdots, t_{Nq}\}$. Set T is randomly separated into N disjoint subsets $T_1, \cdots, T_N$, where there are q secrets in each subset. Then, a subset $\widetilde{T}$ of s secrets is randomly sent to the cloud server. Next, $T - \widetilde{T}$ is again divided into N disjoint subsets $T'_1, \cdots, T'_N$. Each client i receives T_i and T'_i.

Encryption: At each round t, each client generates key $k_i = (\sum_{j \in T_i} H(f_j(t)) - \sum_{j \in T'_i} H(f_j(t)))$ mod M, where $M = 2^{\lceil \log_2(NC) \rceil}$ and C is the maximum value of gradients. As suggested in [12, 13], inner function $f_j(t)$ could be implemented as the HMAC of

t with j as the key to save operation time. Moreover, function H could be formulated in a way that truncates the output of $f_j(t)$ into shorter bits with length of $\log_2 M$ and uses exclusive-OR to compensate the bits of the intermediate output. Each client finally encrypts his own gradients by computing $\overline{g}_i = (g_i + k_i) \mod M$.

Decryption: At each round t, the cloud sever generates key $k_0 = (\sum_{j \in \widetilde{T}} H(f_j(t))) \mod M$, which is used to decrypt the sum S by computing $S = (\sum_{i=1}^{N} g_i - k_0) \mod M$.

Achieving differential privacy for update

Since the accurate sum S of gradients may leak clients' privacy, we adopt differential privacy to add noise to the gradients on the client side so that the cloud server would only learn the noisy sum. Also the noisy sum will not change much whether a specific client participates or not.

In particular, first each client clips the gradients to have bounded sensitivity Δ. The clipping function we use is as follows:

$$Clip(g_i) = \min(1, \frac{C}{||g_i||}) g_i \tag{4.5}$$

where C is the desired gradient bound.

Next, noise r_i is added to each client's gradients such that

$$\overline{g}_i = Clip(g_i) + r_i \tag{4.6}$$

Here, if the noise r_i is too large (e.g. making each clients' gradients differentially private), the cloud server would collect massive noise in the gradient summation, which degrades the performance of trained model. If the noise r_i is too small, the summation of gradients may not be fully protected against privacy leakage. Hence, considering β fraction

compromised clients may contribute wrong noise to cloud server, we set

$$r_i = \frac{Lap(C/\epsilon)}{(1-\beta)p} \tag{4.7}$$

where ϵ is the privacy budget, and p is the total number of clients at each iteration. In this way, clients would *collectively* contribute enough noise to achieve differential privacy in the summation of gradients on the cloud server side.

By following [14], privacy bugdet under Laplace mechanism can be estimated by adopting moments accountant mechanism. In particular, for laplace mechanism with $x \in Lap(\frac{\Delta f}{\epsilon})$, privacy loss function $L(x)$ can be formulated as follows:

$$L(x) = \begin{cases} -\epsilon & \text{if } x > \Delta f \\ -\dfrac{\epsilon}{\Delta f}(2x - \Delta f) & \text{otherwise} \end{cases} \tag{4.8}$$

Let $\Delta f = C$, the λ-th moment function $\alpha_L(\lambda)$, which denotes the log of the moment generating function evaluated at λ, would be obtained as:

$$\alpha_L(\lambda) = \log[\frac{\lambda+1}{2\lambda+1}e^{\lambda\epsilon} + \frac{\lambda}{2\lambda+1}e^{(-\epsilon(\lambda+1))}] \tag{4.9}$$

According to Tail bound theorem in [14], the whole algorithm achieves $(\epsilon, \delta)-$differential privacy by solving the following tail bound equation:

$$\delta = \min_{\lambda} \exp[\alpha_L(\lambda) - \lambda\epsilon] \tag{4.10}$$

Thus, the moments accountant mechanism would output the accumulated privacy cost when the training process terminates.

Algorithm 5 Communication-efficient differentially-private federated learning

Input: Database $D_1, D_2, \cdots, D_N$, total iterations T, mini-batch size B, privacy budget ϵ_1, ϵ_2, learning rate η, number of gradients M, bound C

Output: $\boldsymbol{\theta}^T$

Initialize global parameters and gradients $\overline{\mathbf{g}}^0$
for each iteration $t = 1, 2, \cdots, T$ **do**
  for all client $i = 1, 2, \cdots, N$ in parallel **do**
    Client i receives current global model parameters
    Client i computes the gradients $\mathbf{g}_i^t$ based on his own dataset
    if $\lambda(\mathbf{g}_i^t, \overline{\mathbf{g}}^{t-1}) + Lap(\frac{4M}{\epsilon_1}) \geq \omega_\lambda^t + Lap(\frac{2M}{\epsilon_1})$ or $|\mathbf{g}_i^t| + Lap(\frac{8MC}{\epsilon_1}) \geq \omega_m^t + Lap(\frac{4MC}{\epsilon_1})$ according to Eq. 4.3 and 4.4 **then**
      Client i sends notification to cloud server
    end if
  end for
  The cloud server waits for notifications from all clients and randomly selects p clients at iteration t
  if p clients are selected **then**
    The key dealer runs the setup process to assign secrets to the server and the p clients
    for all client $i = 1, 2, \cdots, p$ in parallel **do**
      Clip gradients $\mathbf{g}_i^t = \min(1, \frac{C}{||\mathbf{g}_i^t||})\mathbf{g}_i^t$ according to Eq. 4.5
      Add Laplacian noise to gradients $\overline{\mathbf{g}}_i^t = \mathbf{g}_i^t + \frac{Lap(C/\epsilon_2)}{(1-\beta)p}$ based on Eq. 4.7
      Encrypt the noisy gradients and upload to cloud server according to section 4.3.4
    end for
    The cloud server decrypts the noisy sum with its key k_0
    Global parameters are updated by: $\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^t - \eta[\frac{1}{pB}(\sum_{i=1}^{p} \mathbf{g}_i^t + Lap(C/\epsilon_2))]$
  else if less than p clients are received at iteration t **then**
    The cloud server performs no global updates and continues to iteration $t+1$. Each client updates his parameters locally for iteration t
  end if
end for
return $\boldsymbol{\theta}^T$ and accumulated privacy cost (ϵ, δ)

To summarize, after the initialization step, all clients whose gradients align well with global gradient or have large norm magnitude would send cloud server a notification. The cloud server waits for notifications from all clients and randomly selects a subset of clients at each iteration to upload their gradients through gradient clipping and encryption uploading steps. Then the cloud server would decrypt the noisy sum with its key and update the global parameters. If there are less than expected clients at the iteration, the global gradient updating process would abort and proceed to next iteration. All the clients would update their parameters locally at this iteration. The entire communication efficient differential privacy algorithm is summarized in Algorithm 5.

LEMMA 1. Let's set global sensitivity $\Delta = 2C$, Algorithm 5 achieves ϵ_2-differential privacy for cloud aggregation.

Proof 1. Let two datasets D and D' be neighboring datasets. Without loss of generality, assume D and D' differ in the last tuple $x_n(x'_n)$. Aggregation in the cloud server is written as follows:

$$\hat{\mathbf{g}} = \sum_{i=1}^{p} \mathbf{g}_i^t + Lap(C/\epsilon_2) \tag{4.11}$$

So we have:

$$
\begin{aligned}
\frac{Pr(\mathbf{S}(D)=z)}{Pr(\mathbf{S}(D')=z)} &= \frac{\prod_{i=1}^{n}\exp(\frac{\epsilon_2\|\frac{1}{|D|}\sum_{x_i\in D}g(x_i)-\hat{g}\|}{\Delta})}{\prod_{i=1}^{n}\exp(\frac{\epsilon_2\|\frac{1}{|D'|}\sum_{x'_i\in D'}g(x'_i)-\hat{g}\|}{\Delta})} \\
&\leq \prod_{i=1}^{n}\exp(\frac{\epsilon_2 \parallel \sum_{x_i\in D}g(x_i)-\sum_{x'_i\in D'}g(x'_i) \parallel}{|D|\Delta}) \\
&\leq \prod_{i=1}^{n}\exp(\frac{\epsilon_2 \parallel g(x_n)-g(x'_n) \parallel}{|D|\Delta}) \\
&\leq \prod_{i=1}^{n}\exp(\frac{2\epsilon_2 C}{|D|\Delta}) \\
&\leq \exp(\frac{\sum_{i=1}^{n}2\epsilon_2 C}{|D|\Delta}) \\
&= \exp(\epsilon_2)
\end{aligned}
\tag{4.12}
$$

This concludes the proof. □

4.4 Performance Evaluation

4.4.1 Benchmark Frameworks

In this section, the effectiveness of proposed algorithm will be evaluated. We implement the following four SGD-based frameworks for comparison.

1) *Vanilla FL*[15]: This is the conventional, no-privacy framework.

2) *DSSGD*[9]: Under this scheme, differential privacy noise is added into selected gradients. The parameters (e.g. batch size, learning rate) of DSSGD are tuned to its best performance and the hyperparameters are set as follows: gradient upload ratio $\theta_u = 0.5$, gradient download ratio $\theta_d = 1$, gradient selection threshold $\tau = 0.0001$, and gradient bound $\gamma = 0.001$.

3) *EPFL*[16]: Under this scheme, differential privacy noise is added into gradients.

Then additively homomorphic encryption is adopted to encrypt noisy gradients before uploading to cloud server.

4) *Standalone framework:* In this framework, clients individually train local models using standard SGD without any collaboration, which are susceptible to being trapped at local optima.

4.4.2 Dataset and Experimental Setup

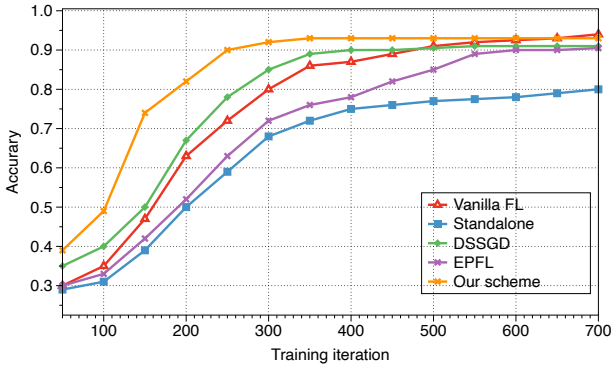
We experiment on two benchmark datasets: MNIST [17] and SVHN [18]. There are 70,000 data records in the MNIST dataset, out of which 60,000 records are training records and the remaining are testing records. The size of each record is 32×32, and it is grey-level image with digits centered and ranging from 0 to 9. SVHN is another well-known dataset which shows house numbers in different street views. There are 600,000 32×32 color images in SVHN dataset. We use 100,000 as training examples and 10,000 as testing examples.

We use the accuracy of classification to evaluate the performance of different frameworks. All the samples are sorted by their digit labels and then each client receives same amount of examples (3,000 examples for MNIST, and 5,000 examples for SVHN), which simulates a non-IID data distribution.

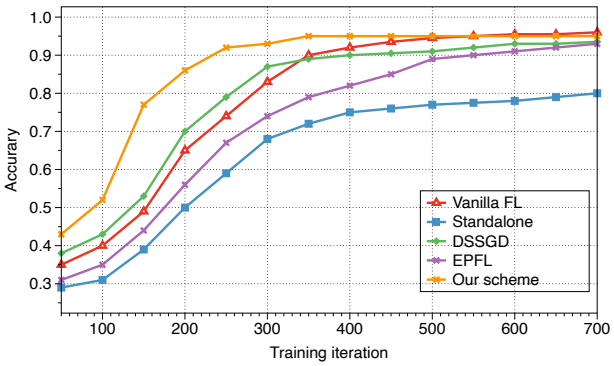
In this experiment, one of the most popular neural network architectures is implemented, *convolutional neural network* (CNN). CNN framework is built with two convolutional layers, which is followed by a Sigmoid layer and a max pooling layer for each convolutional layer.

The architecture is implemented using Pytorch [19] framework. We set the learning rate as 0.01 and the mini-batch size as 16. For weights initialization, we draw random value from normal distribution with 0 mean and 1 standard deviation.

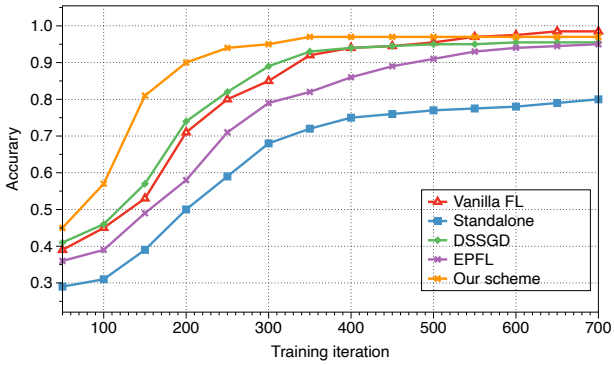
For MNIST dataset, we set clipping bound $C = 1$, and $\delta = 10^{-5}$. For SVHN dataset,



(a) 30 client nodes



(b) 60 client nodes



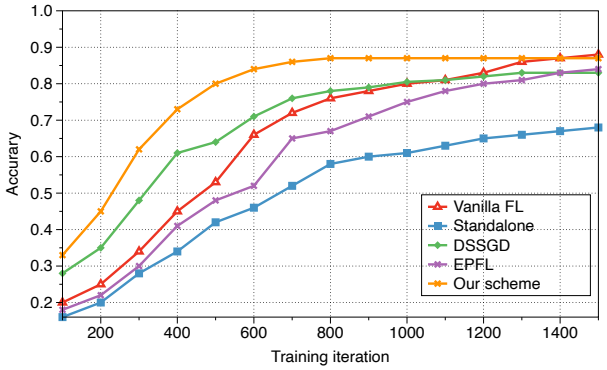
(c) 90 client nodes

Figure 4.2: Model accuracy for different frameworks (MNIST)

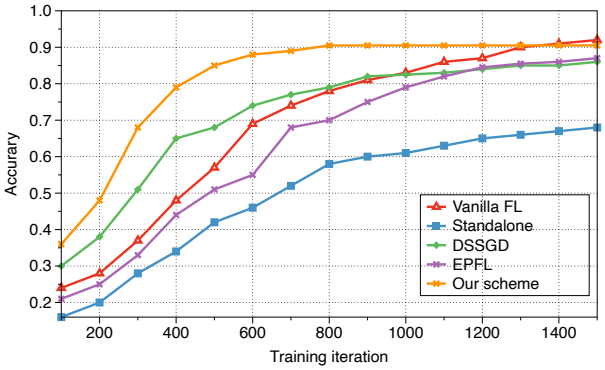
we set clipping bound $C = 0.5$, and $\delta = 10^{-5}$. $\epsilon_1 = \epsilon_2 = 1/2\epsilon$. The accumulated privacy budget ϵ for each setting is computed using the privacy moments accounting method [14]. The experiments were set up with different clients to simulate federated learning setting. The initial alignment threshold ω_λ is set as 0.5. The initial value of ω_m is set as 10%. Below, we evaluate the performance of our framework in different settings.

4.4.3 Communication Overhead

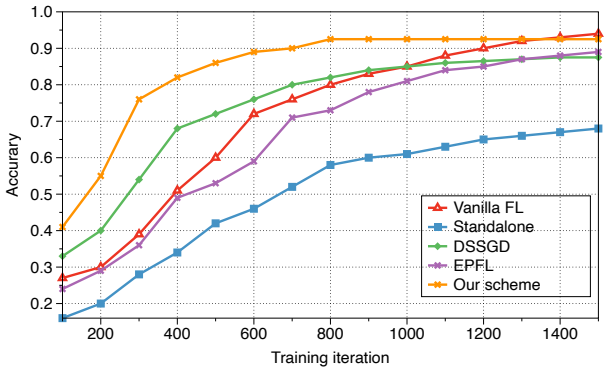
In this section, communication overhead is measured among different frameworks. Here we set an intermediate value of $\epsilon = 10$ and upper bound for δ as 10^{-5}. As we can see from Fig. 4.2 and 4.3, generally our scheme outperforms DSSGD, EPFL and standalone frameworks under different settings and datasets. With the same training accuracy, the



(a) 30 client nodes



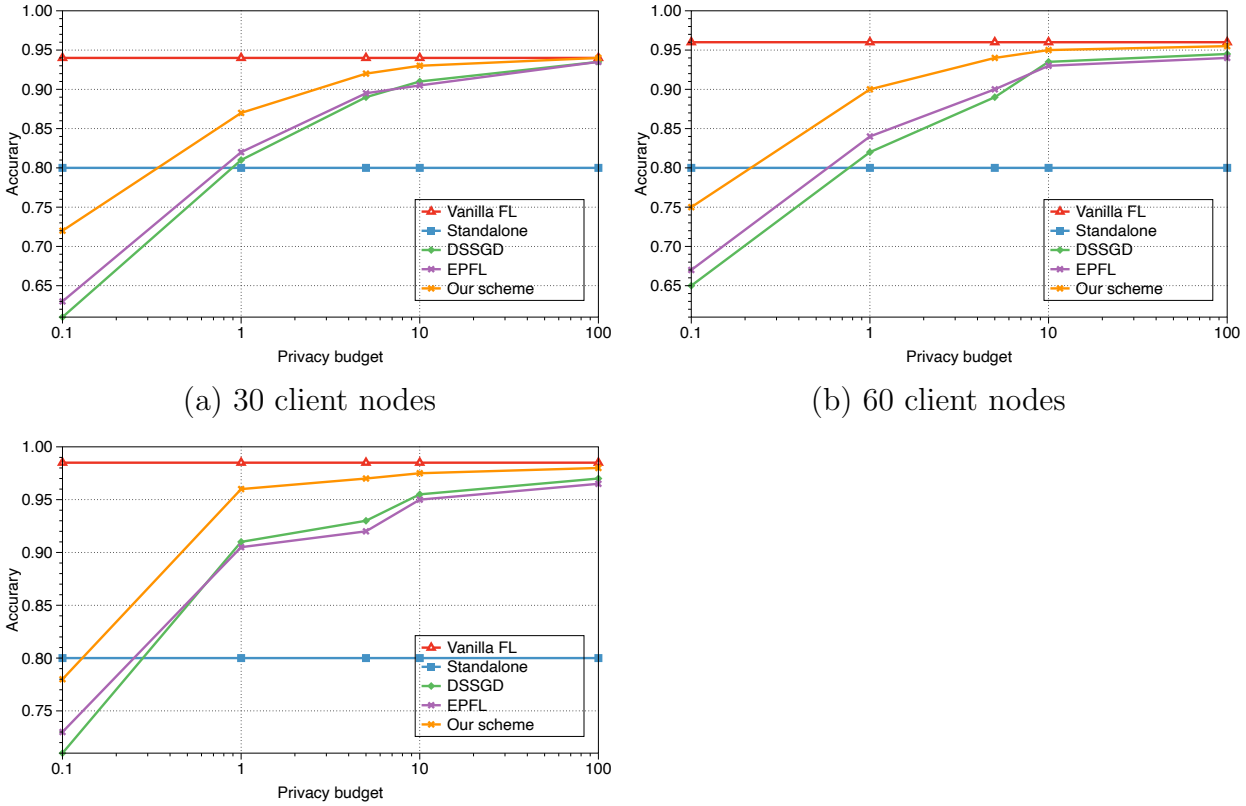
(b) 60 client nodes



(c) 90 client nodes

Figure 4.3: Model accuracy for different frameworks (SVHN)

performance of our scheme needs much less training iterations than DSSGD, EPFL and Vanilla FL. For example, under 30 client nodes scenario with MNIST dataset, when training accuracy reaches 80%, our scheme needs only 180 iterations under 30 clients setting, while DSSGD needs 44% more iterations, Vanilla FL needs 66% more iterations and EPFL needs 133% more iterations, which shows the effectiveness of uploading aligned gradients in our scheme. For SVHN dataset, the comparison results are also similar to performance with MNIST dataset. As the total number of clients increases, generally we achieve better performance results. This is mainly due to more data is available to the model at each iteration.



(a) 30 client nodes

(b) 60 client nodes

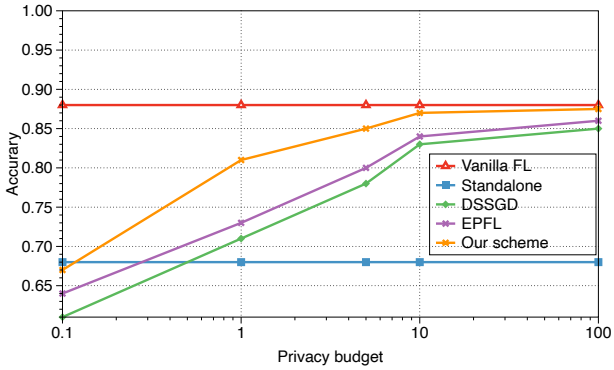
(c) 90 client nodes

Figure 4.4: Accuracy of different frameworks with varying privacy budget (MNIST)

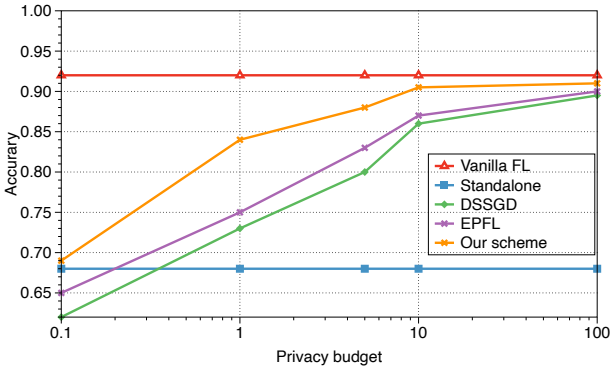
4.4.4 Privacy Budget

In this experiment, we will evaluate the tradeoff between model accuracy and privacy budget ϵ. Specifically, the alignment threshold $\omega_\lambda = 0.8, \omega_m = 0.1$. We vary the privacy budget from $0.1, \cdots, 100$ with δ as 10^{-5}.

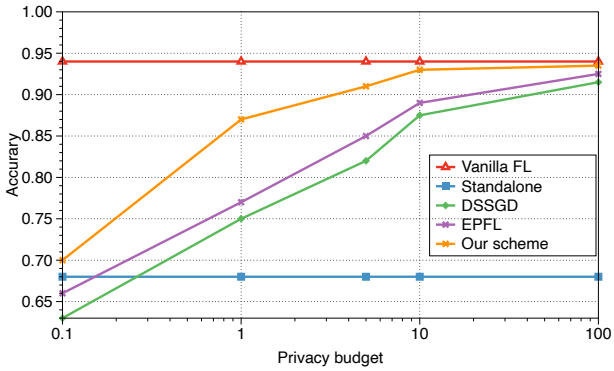
As we can see from Fig. 4.4 and 4.5, our proposed scheme could achieve comparable accuracy with vanilla FL framework with modest privacy budget. For example, when $\epsilon = 5, 10, N = 30$ in MNIST dataset, our scheme achieves 92.5% and 93.5% accuracy respectively, which is comparable to that of vanilla FL (i.e. 94%). Moreover, our scheme outperforms EPFL, DSSGD and standalone frameworks with different privacy budget. This is mainly due to much less noise is added on the client side during training, hence the model performance would not degrade much. With more clients participating in the federated learning, the



(a) 30 client nodes



(b) 60 client nodes



(c) 90 client nodes

Figure 4.5: Accuracy of different frameworks with varying privacy budget (SVHN)

model accuracy can be improved even with stronger privacy guarantee (e.g. $\epsilon = 0.1, 1$).

4.4.5 Scalability

In this section, we evaluate the scalability of the proposed algorithm with different number of clients, which is illustrated in Fig. 4.6. A small $\epsilon = 0.5$ is adopted, $\delta = 10^{-5}$, and the number of clients ranges from 30 to 150. As shown in the Fig. 4.6, the model accuracy improves with more clients participating in federated learning, which is due to the increasing availability of training data.

In summary, our proposed scheme achieves the best tradeoff between communication cost and privacy guarantee of clients' data and scales well.

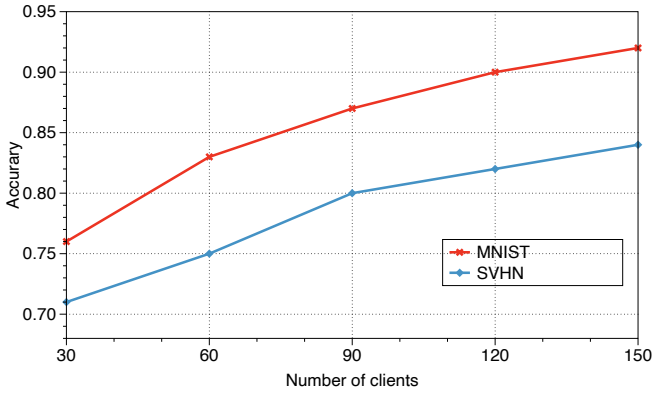


Figure 4.6: Federated learning with different clients ($\epsilon = 0.5, \delta = 10^{-5}$)

4.5 Related Work

This section provides the background of efficient federated learning algorithms, as well as privacy-preserving federated learning approaches.

4.5.1 Efficient Federated Learning

Most efficient distributed deep learning works can be classified into two categories. The first category is quantization, which reduces the precision of parameters. Vanhoucke et al. [20] proposed to leverage fixed-point instructions to facilitate the training of neural networks. Hubara et al. [21] demonstrated to significantly reduce run-time communication overhead during training with short bit weights and activations. Shen et al. [22] also proposed to use second order Hessian information for quantizing BERT models to ultra low precision. Suresh et al. [23] applied a structured random rotation before quantization to further reduce the less error of training objectives. Sun et al. [24] proposed Lazily Aggregated Quantized gradient (LAQ) to reduce transmitted bits as well as communication rounds.

The other category is sparsification. Strom et al. [25] proposed to select important gradients based on a constant threshold, and only uploads ones greater than the threshold. Mills et al. [26] proposed to leverage a distributed form of Adam optimisation for reducing communcaitoon overhead. Teerapittayanon et al. [27] proposed to generate precise feature

extraction by dissecting DNN frameworks into two parts and improve performance in the cloud. However, end users are involved in the iterative training phase, where training passes gradients through devices, resulting in high communication overhead. Ivkin et al. [28] demonstrated to communicate gradient sketches during training to reduce communication cost. Nikoli et al. [29] proposed to choose the gradients based on threshold and then encode it with lower-bits. Shi et al. [30] developed an optimal merged gradient sparsification algorithm, which merges gradients in different layers to further accelerate the training. Chai et al. [31] presented asynchronous training tiers with each client holding Non-i.i.d. dataset, which supports synchronous and asynchronous tiers training to accelerate model training.

Our work belongs to the line of sparsification, but selecting gradients based on alignment with global gradient and gradient magnitude has received little attention.

4.5.2 Privacy-preserving Federated Learning

Existing works on privacy-preserving distributed learning mostly utilize either differential privacy mechanism, secure multiparty computation or homomorphic encryption. For differential privacy, Pathak et al. [4] introduced a DP-based global classifier by aggregating the locally trained classifiers. Sun et al. [32] proposed to apply Noise-Free Differential Privacy (NFDP) mechanism into a federated learning framework. Chen et al. [33] perturbed local embedding to ensure the differential privacy of local information. Choudhury et al. [34] studied applying differential privacy to protect sensitive healthcare data. Triastcyn and Faltings [35] improved the process of recording privacy budgets in distributed learning at different training stages by adopting Bayesian privacy accounting method. Zhao et al. [36] proposed to efficiently improve performance of crowdsourcing applications under federated learning based on local differential privacy (LDP).

Similarly, secure multiparty computation aims to jointly compute a function over dis-

tributed participants data while ensuring data privacy. Xu et al. [37] proposed to employ an SMC protocol based on functional encryption to account for participants dropping out during training. Bonawitz et al. [38] adopted using secret sharing that enables authenticated encryption. Byrd and Polychroniadou [39] constructed a federated learning system by applying differential privacy on top of MPC to learn on financial institute datasets.

A number of works have been proposed to adopt homomorphic encryption for federated learning tasks. Xu et al. [40] proposed to protect users' data privacy (i.e. local gradients) with double-masking scheme during distributed learning process. The users can also verify the aggregated results from cloud server by testing the Proof. Liu et al. [41] built an homomorphic encryption based framework to train federated transfer learning setting. Hardy et al. [42] adopted additively homomorphic scheme to train federated logistic regression model. Liu et al. [43] proposed utilizing partial homomorphic encryption method to encrypt model parameters for optimizing training convergence process. Zhang et al. [44] proposed to adopt a large integer for encoding and concatenating gradient batches. Then the integer would be encrypted for transmission. They also presented to efficiently perform element-wise aggregation in the encoded gradient batches by applying advanced quantization schemes. Gradient clipping protocol was also proposed to help bound the gradient range. However, efficient federated learning with both homomorphic encryption and differential privacy has not been studied well.

4.6 Summary

In this chapter, we have investigated efficient federated learning with differential privacy. In particular, each client uploads his gradients based on the efficient gradient upload scheme. Then, the selected gradients would be perturbed and encrypted before transmitting to cloud server for averaging. The cloud server would learn the noisy sum but

nothing else during the training process. In addition, the noisy sum would not change much whether a specific client participates or not. Extensive experimental results show that the proposed framework can achieve high training accuracy with strong privacy guarantee while significantly reduce communication cost and scales well.

Bibliography

[1] S. B. Kotsiantis, I. Zaharakis, and P. Pintelas, "Supervised machine learning: A review of classification techniques," *Emerging artificial intelligence applications in computer engineering*, vol. 160, pp. 3–24, 2007.

[2] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016.

[3] O. Ohrimenko, F. Schuster, C. Fournet, A. Mehta, S. Nowozin, K. Vaswani, and M. Costa, "Oblivious multi-party machine learning on trusted processors," in *25th {USENIX} Security Symposium ({USENIX} Security 16)*, pp. 619–636, 2016.

[4] M. A. Pathak, S. Rane, and B. Raj, "Multiparty differential privacy via aggregation of locally trained classifiers.," in *NIPS*, pp. 1876–1884, Citeseer, 2010.

[5] B. Hitaj, G. Ateniese, and F. Perez-Cruz, "Deep models under the gan: information leakage from collaborative deep learning," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pp. 603–618, 2017.

[6] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of COMPSTAT'2010*, pp. 177–186, Springer, 2010.

[7] O. Dekel, R. Gilad-Bachrach, O. Shamir, and L. Xiao, "Optimal distributed online prediction using mini-batches," *Journal of Machine Learning Research*, vol. 13, no. Jan, pp. 165–202, 2012.

[8] D. van Esch, E. Sarbar, T. Lucassen, J. O'Brien, T. Breiner, M. Prasad, E. Crew, C. Nguyen, and F. Beaufays, "Writing across the world's languages: Deep internationalization for gboard, the google keyboard," *arXiv preprint arXiv:1912.01218*, 2019.

[9] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, pp. 1310–1321, ACM, 2015.

[10] K. Hsieh, A. Harlap, N. Vijaykumar, D. Konomis, G. R. Ganger, P. B. Gibbons, and O. Mutlu, "Gaia: Geo-distributed machine learning approaching {LAN} speeds," in *14th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 17)*, pp. 629–647, 2017.

[11] C. Dwork, A. Roth, *et al.*, "The algorithmic foundations of differential privacy.," *Foundations and Trends in Theoretical Computer Science*, vol. 9, no. 3-4, pp. 211–407, 2014.

[12] Q. Li and G. Cao, "Efficient privacy-preserving stream aggregation in mobile sensing with low aggregation error," in *International Symposium on Privacy Enhancing Technologies Symposium*, pp. 60–81, Springer, 2013.

[13] C. Castelluccia, A. C. Chan, E. Mykletun, and G. Tsudik, "Efficient and provably secure aggregation of encrypted data in wireless sensor networks," *ACM Transactions on Sensor Networks (TOSN)*, vol. 5, no. 3, pp. 1–36, 2009.

[14] M. Park, J. Foulds, K. Choudhary, and M. Welling, "Dp-em: Differentially private expectation maximization," in *Artificial Intelligence and Statistics*, pp. 896–904, PMLR, 2017.

[15] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial Intelligence and Statistics*, pp. 1273–1282, PMLR, 2017.

[16] M. Hao, H. Li, G. Xu, S. Liu, and H. Yang, "Towards efficient and privacy-preserving federated deep learning," in *ICC 2019-2019 IEEE International Conference on Communications (ICC)*, pp. 1–6, IEEE, 2019.

[17] L. Deng, "The mnist database of handwritten digit images for machine learning research [best of the web]," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[18] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," 2011.

[19] N. Ketkar, "Introduction to pytorch," in *Deep learning with python*, pp. 195–208, Springer, 2017.

[20] V. Vanhoucke, A. Senior, and M. Z. Mao, "Improving the speed of neural networks on cpus," 2011.

[21] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Quantized neural networks: Training neural networks with low precision weights and activations," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 6869–6898, 2017.

[22] S. Shen, Z. Dong, J. Ye, L. Ma, Z. Yao, A. Gholami, M. W. Mahoney, and K. Keutzer, "Q-bert: Hessian based ultra low precision quantization of bert," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, pp. 8815–8821, 2020.

[23] A. T. Suresh, X. Y. Felix, S. Kumar, and H. B. McMahan, "Distributed mean estimation with limited communication," in *International Conference on Machine Learning*, pp. 3329–3337, PMLR, 2017.

[24] J. Sun, T. Chen, G. B. Giannakis, and Z. Yang, "Communication-efficient distributed learning via lazily aggregated quantized gradients," *arXiv preprint arXiv:1909.07588*, 2019.

[25] N. Strom, "Scalable distributed dnn training using commodity gpu cloud computing," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[26] J. Mills, J. Hu, and G. Min, "Communication-efficient federated learning for wireless edge intelligence in iot," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 5986–5994, 2019.

[27] S. Teerapittayanon, B. McDanel, and H.-T. Kung, "Distributed deep neural networks over the cloud, the edge and end devices," in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, pp. 328–339, IEEE, 2017.

[28] N. Ivkin, D. Rothchild, E. Ullah, V. Braverman, I. Stoica, and R. Arora, "Communication-efficient distributed sgd with sketching," *arXiv preprint arXiv:1903.04488*, 2019.

[29] N. Dryden, T. Moon, S. A. Jacobs, and B. Van Essen, "Communication quantization for data-parallel training of deep neural networks," in *2016 2nd Workshop on Machine Learning in HPC Environments (MLHPC)*, pp. 1–8, IEEE, 2016.

[30] S. Shi, Q. Wang, X. Chu, B. Li, Y. Qin, R. Liu, and X. Zhao, "Communication-efficient distributed deep learning with merged gradient sparsification on gpus," in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*, pp. 406–415, IEEE, 2020.

[31] Z. Chai, Y. Chen, L. Zhao, Y. Cheng, and H. Rangwala, "Fedat: A communication-efficient federated learning method with asynchronous tiers under non-iid data," *arXiv preprint arXiv:2010.05958*, 2020.

[32] L. Sun and L. Lyu, "Federated model distillation with noise-free differential privacy," *arXiv preprint arXiv:2009.05537*, 2020.

[33] T. Chen, X. Jin, Y. Sun, and W. Yin, "Vafl: a method of vertical asynchronous federated learning," *arXiv preprint arXiv:2007.06081*, 2020.

[34] O. Choudhury, A. Gkoulalas-Divanis, T. Salonidis, I. Sylla, Y. Park, G. Hsu, and A. Das, "Differential privacy-enabled federated learning for sensitive health data," *arXiv preprint arXiv:1910.02578*, 2019.

[35] A. Triastcyn and B. Faltings, "Federated learning with bayesian differential privacy," in *2019 IEEE International Conference on Big Data (Big Data)*, pp. 2587–2596, IEEE, 2019.

[36] Y. Zhao, J. Zhao, M. Yang, T. Wang, N. Wang, L. Lyu, D. Niyato, and K.-Y. Lam, "Local differential privacy based federated learning for internet of things," *IEEE Internet of Things Journal*, 2020.

[37] R. Xu, N. Baracaldo, Y. Zhou, A. Anwar, and H. Ludwig, "Hybridalpha: An efficient approach for privacy-preserving federated learning," in *Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security*, pp. 13–23, 2019.

[38] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for privacy-preserving machine learning," in *proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1175–1191, 2017.

[39] D. Byrd and A. Polychroniadou, "Differentially private secure multi-party computation for federated learning in financial applications," *arXiv preprint arXiv:2010.05867*, 2020.

[40] G. Xu, H. Li, S. Liu, K. Yang, and X. Lin, "Verifynet: Secure and verifiable federated learning," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 911–926, 2019.

[41] Y. Liu, Y. Kang, C. Xing, T. Chen, and Q. Yang, "A secure federated transfer learning framework," *IEEE Intelligent Systems*, vol. 35, no. 4, pp. 70–82, 2020.

[42] S. Hardy, W. Henecka, H. Ivey-Law, R. Nock, G. Patrini, G. Smith, and B. Thorne, "Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption," *arXiv preprint arXiv:1711.10677*, 2017.

[43] C. Liu, S. Chakraborty, and D. Verma, "Secure model fusion for distributed learning using partial homomorphic encryption," in *Policy-Based Autonomic Data Governance*, pp. 154–179, Springer, 2019.

[44] C. Zhang, S. Li, J. Xia, W. Wang, F. Yan, and Y. Liu, "Batchcrypt: Efficient homomorphic encryption for cross-silo federated learning," in *2020 {USENIX} Annual Technical Conference ({USENIX}{ATC} 20)*, pp. 493–506, 2020.

5 Conclusions and Future Work

5.1 Conclusions

In this dissertation, we proposed a set of systems to protect users' privacy in cloud-assisted data analytics.

In Chapter 2, we proposed a secure, efficient, and verifiable outsourcing protocol for geometric programming, which has wide applications in data analysis. In particular, a secure and efficient transformation scheme is employed to protect the data privacy. We apply the gradient projection method to solve the encrypted geometric programming problem in the cloud side. Experiments were conducted on both Amazon Elastic Compute Cloud (EC2) and a laptop to evaluate performance of the designed outsourcing protocol, and the results showed the feasibility and efficiency of the proposed algorithm.

In Chapter 3, we designed a differentially private framework to train logistic regression models out of distributed data sources (e.g., individual users and organizations). To achieve high learning accuracy while maintaining privacy, our solution considers the relevance between input data features and the model output when generating noises. In particular, at each data owner, more noise is added to the coefficients of the objective polynomial form that are less relevant to the local model output, and less noise to those that are more relevant. When the local parameters are uploaded to a cloud server for aggregation, the server uses an evaluation dataset to assess the data quality of clients, and then selects the model parameters of a subset of clients into aggregation based on the data quality of clients while protecting clients privacy. Extensive experimental results showed that the proposed framework can achieve high classification accuracy while protecting privacy and being robust against low-quality data.

In Chapter 4, we proposed an efficient privacy-preserving federated learning system that enables multiple clients to collaboratively train their neural network models without revealing their own dataset. In particular, each user selectively chooses the gradients to upload based on the tendency of the global model convergence and gradient magnitude. Moreover, clients first perturb the gradients uploading to cloud server by adding Laplacian noise. Then, the noisy gradients are encrypted with homomorphic encryption before uploading to cloud server. Finally, the cloud server decrypts the sum of the noisy gradients and updates the global model parameters during the training process. Extensive experiments on benchmark dataset show that the efficient privacy-preserving deep learning scheme significantly reduces communication overhead, achieves minimal accuracy loss with strong data privacy and scales well.

5.2 Future Work

This dissertation proposed several systems to provide privacy protection for cloud-assisted data analytics. Besides collecting more datasets to better validate the proposed systems, there are still many other issues that deserve further exploration. In the following, we discuss two future research directions.

- **Privacy-Preserving Cloud-Assisted Data Analytics with Fair Exchange:** Although we proposed a secure and efficient outsourcing scheme for large-scale geometric programming, we have not addressed the fair exchange issue in the system. In particular, if the cloud server performs computation tasks before user paying the service fee, the user might not pay after receiving the results. If the user pays the service fee first, the cloud server might not execute the computation and return random and invalid results. Some work have been done to address the fair exchange issue [1, 2, 3, 4], but their schemes have not been adopted in secure outsourcing of large scale mathematical

optimizations. Therefore, it is necessary to design a privacy-preserving outsourcing system with fair exchange.

- **Privacy-Preserving Distributed Learning with Heterogeneous Architectures:** Even though we have designed privacy-preserving distributed logistic regression and efficient federated learning systems to protect user privacy during training, the current design can only learn the same model across all the users. Since different users may need different learning models, it would be interesting to study how to extend federated learning to collaboratively train models with heterogeneous architectures.

Bibliography

[1] H. Shafagh, L. Burkhalter, A. Hithnawi, and S. Duquennoy, "Towards blockchain-based auditable storage and sharing of iot data," in *Proceedings of the 2017 on Cloud Computing Security Workshop*, pp. 45–50, 2017.

[2] K. Christidis and M. Devetsikiotis, "Blockchains and smart contracts for the internet of things," *Ieee Access*, vol. 4, pp. 2292–2303, 2016.

[3] A. B. Kurtulmus and K. Daniel, "Trustless machine learning contracts; evaluating and exchanging machine learning models on the ethereum blockchain," *arXiv preprint arXiv:1802.10185*, 2018.

[4] C. Lin, D. He, X. Huang, X. Xie, and K.-K. R. Choo, "Blockchain-based system for secure outsourcing of bilinear pairings," *Information Sciences*, vol. 527, pp. 590–601, 2020.